Numerical modeling of large scales and long time

Jérémy Veysset

ParisTech
INSTITUT DES SCIENCES ET TECHNOLOGIES
PARIS INSTITUTE OF TECHNOLOGY

MINES
ParisTech

École doctorale n° 364 : Sciences Fondamentales et Appliquées

Doctorat ParisTech

THÈSE

pour obtenir le grade de docteur délivré par

l'école Nationale Supérieure des Mines de Paris

Spécialité doctorale "Mécanique Numérique"

présentée et soutenue publiquement par

M. Jérémy Veysset

le 29 Septembre 2014

Simulation des grands espaces et des temps longs

Numerical modeling of large scales and long time

Directeurs de thèse : **Elie Hachem, Thierry Coupez**

Jury

M. Marco Picasso,	Professeur, Ecole Polytechnique Fédérale de Lausanne	Rapporteur
M. Frédéric Hecht,	Professeur, Université Pierre et Marie Curie	Rapporteur
M. Pedro Diez,	Professeur, Universitat Politècnica de Catalunya BarcelonaTech	Examinateur
M. Bernard Mourrain,	Directeur de Recherche, INRIA Sophia Antipolis	Examinateur
M. Elie Hachem,	Maître assistant, HDR, Mines ParisTech	Examinateur
M. Thierry Coupez,	Professeur, Ecole Centrale de Nantes	Examinateur
Mme. Isabelle Poitrault,	Industeel, ArcelorMittal	Examinateur

MINES ParisTech
Centre de Mise Forme des Matériaux (CEMEF)
UMR CNRS 7635, F-06904 Sophia Antipolis, France

THÈSE

Abstract

Fluid-Structure Interaction (FSI) describes a wide variety of industrial problems arising in mechanical engineering, civil engineering and biomechanics. In spite of the available computer performance and the actual maturity of computational fluid dynamics and computational structural dynamics, several key issues still prevent accurate FSI simulations.

Two main approaches for the simulation of FSI problems are still gaining attention lately: partitioned and monolithic approaches. Results in the literature show that the partitioned approach is accurate and efficient but some instabilities may occur depending on the ratio of the densities and the complexity of the geometry. Monolithic methods are still of interest due to their capability to treat the interaction of the fluid and the structure using a unified formulation. In fact it makes the build up of a FSI problem easier as the mesh do not have to fit the geometry of the solids and the transfers are treated naturally.

The software Thost has been created based on these analyzes. Thost is a 3D aerothermal numerical software. It has been developed for the numerical simulation of industrial processes like the heating in industrial furnaces as well as quenching. Its target is to model numericaly the thermal history of the industrial pieces in their environment without using any transfer coefficient. However the computational costs are still high and therefore the software is not fully efficient from an industrial point of view to simulate, analyze and improve complex processes. All the work in this PhD thesis has been done to reduce the computational costs and optimize the accuracy of the simulations in Thost based on innovative numerical methods such as dynamic anisotropic mesh adaptation, stabilized finite elements methods and immersing the objects directly from their Computer Aided Design files.

Résumé

L'interaction fluide structure est présente dans beaucoup de problèmes industriels, dans les domaines d'ingénierie mécanique, civile ou biomécanique. Même si les performances informatique s'améliorent considérablement et que les méthodes en mécanique numérique gagnent en maturité, certaines difficultés ne permettent pas encore de réaliser des simulations numériques précises.

Actuellement deux méthodes numériques gagnent en popularité pour la simulation numérique d'interactions fluide structure: la méthode de partitionnement et la méthode monolithique. Des résultats de la littérature montrent que la première est efficace et précise mais qu'elle peut rencontrer des problèmes d'instabilité si les ratios de densité sont élevés ou que les géométries sont complexes. Les méthodes d'immersion sont de plus en plus utilisées par la communauté scientifique. Différentes approches ont été développée, dont la Méthode d'Immersion de Volume. Cette méthode permet de faciliter la mise en place des calculs. Ainsi il n'est pas nécessaire de construire des maillages concordant avec la géométrie des objets, et le couplage entre les fluides et les solides se fait naturellement.

C'est sur cette analyse qu'à été développé le logiciel Thost. Il permet de simuler des procédés industriels tels que le chauffage de pièces métalliques dans les fours industriels ou la trempe sans caractériser expérimentalement des coefficients de transfert. Le but d'un tel logiciel est de permettre une meilleur compréhension des procédés et ainsi de les optimiser. Cependant les coûts de calcul restant élevés, le but de la thèse est de les diminuer en s'appuyant sur des méthodes numériques innovantes tels que le l'adaptation dynamique de maillage anisotrope, des méthodes éléments finis stabilisées ou l'immersion direct des objets à partir de la Conception Assistée par Ordinateur.

Contents

Part I

Part A

Chapter 1

General introduction

Fluid-Structure Interaction (FSI) describes a wide variety of industrial problems arising in mechanical engineering, civil engineering and biomechanics. In spite of the available computer performance and the actual maturity of computational fluid dynamics (CFD) and computational structural dynamics (CSD), several key issues still prevent accurate FSI simulations.

Two main approaches for the simulation of FSI problems are still gaining attention lately: partitioned and monolithic approaches. The partitioned approaches allow the use of a specific solver for each domain. The fluid and the structure equations are alternatively integrated in time and the interface conditions are enforced asynchronously. The difficulty remains in transferring the informations between the codes. Different schemes, weakly or strongly coupled, are used to ensure the coupling between the two phases. The weakly coupled approach requires one solution of either field per time step but it consequently affects the accuracy of the coupling conditions. The strongly coupled version requires sub-iterations [1–6]. Results in the literature show that the approach is accurate and efficient. However, some instabilities may occur depending on the ratio of the densities and the complexity of the geometry [7].

Monolithic methods are still of interest due to their capability to treat the interaction of the fluid and the structure using a unified formulation [8–10]. In this case, there is no need to enforce the continuity at the interface, it is obtained naturally once the structure is immersed in the fluid domain. One unique conservation equation is then used to describe both the solid and the fluid domains.

1.1 Immersed Methods

Immersed methods for FSI are gaining popularity in many scientific and engineering applications. Different approaches can be found such as the embedded boundary method

[11], the immersed Boundary method [12], the fictitious domain [13], the Immersed Volume method [14] and the Cartesian method [15]. All these methods are attractive because they simplify a number of issues in Fluid-Structure applications such as meshing the fluid domain, using of a fully Eulerian algorithm, dealing with problems involving large structural motions and deformations [16] or topological changes [17].

However the use of non-body fitted grids requires a special interface treatment. Indeed recent developments are focusing on issues related to the immersion of a surface mesh for complex 3D geometries, the detection and the intersection algorithms for the interface and also the transmission of boundary conditions between the solid and the fluid regions [18–20].

The software Thost has been created based on these analyzes. Thost is a 3D aerothermal numerical software. It has been developped for the numerical simulation of industrial processes like the heating in industrial furnaces as well as quenching. Its target is to model numericaly the thermal history of the indutrial pieces in their environment without using any transfer coefficient. As the transfer coefficients between the pieces and the surrounding fluids are not easy to collect and quantify and are really case dependent, Thost offers a generic and flexible framework to optimize a large variety of complex indusrial processes. To avoid the evaluation of the transfer coefficients the software uses an immersed method that implies a strong direct fluid-solid coupling. Therefore the treated pieces or the phases of the flow are represented implicitly and taken into account by a signed distance function, or level-set. This simplifies considirebly the definition and the management of the mesh compared to body fitted methods. Moreover, with this kind of method it is much more easier to add, remove or even move objects during the computation as it is not necessary to create and fit the whole geometry to the new configuration. The software has demonstrated its capability for solving turbulent flow problems coupled to conjugated heat transfer [21]. However the computational costs are still high and therefore the software is not fully efficient from an industrial point of view to simulate, analize and improve complex processes. Therefore the target of the project REALisTIC is to reduce the computational costs and optimize the accuracy of the simulations in Thost. This PhD is part of this project.

This software is based on the Cimlib library [22] which has been developed at the CEMEF (Material Forming) laboratory. Cimlib is a fully parallel C++ finite elements library. This library uses the PETSc library [23] to perform the system resolution and is based on a parallel mesher [24]. The parallelism is managed via MPI (Message Passing Interface).

In this work, we thus use the immersed volume method and present its extension. It uses the levelset function to describe the immersed structure. For simple geometries, we resort to the use of analytical functions (i.e. sphere, square, ...). Whereas to compute the distance function for a complex geometry we use its surface, described and discretized by a simplex mesh (a set of triangles for three-dimensional simulations or a set of segments

for two-dimensional simulations). Then we compute the distance from any given points (a node of the computational domain) to the surface mesh. It is clear that in this case, the description of the immersed structure is limited by the quality and the accuracy of the given surface mesh. Therefore, we propose a new immersion technique that simplifies and bypasses the generation of these meshes. It is based on the direct use of Non Uniform Rational B-Splines (NURBS) curves or surfaces, representing simple or complex geometries. We compute the distance function from any point in the fluid mesh to these NURBS, thus representing the immersed solid by the zero iso-value of this function.

Up to now the objects and pieces were immersed in the computational domain and the mesh was adapted all around as an initial step. This provided the guaranty of accurate heat transfers at the fluid-solid interface. Then the problem was solved by the use of stabilized finite element methods. The inconvenience of such an approach is the impossibility to change the computation configuration (as mentionned above move an object for example) because the mesh would not be well adapted at the object interfaces and at the boundary layers. To remedy this drawback we use a dynamic anisotropic mesh adaptation method. Thus the immersion process is looped as described by Figure 1.1.

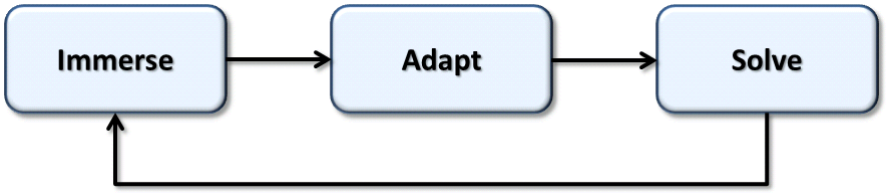


FIGURE 1.1: New computation cycle used to adapt the mesh during the numerical simulation

1.1.1 Immersed Volume Method

The Immersed Volume Method is an interesting tool for computational engineers, in particular for conjugate heat transfer analysis. It can be easily implemented in finite element codes. It allows solving a single set of equations for the whole computational domain and treating different subdomains as a single fluid with variable material properties. This offers a great flexibility to deal with different shapes or to change easily the physical properties for each immersed structure. Therefore, we start by computing the signed distance function of a given geometry to each node of the mesh. Using the zero isovalue of this function, we can easily identify the fluid-solid interface. Consequently, we can apply an anisotropic mesh adaptation at this interface and then mix the thermo-physical properties appropriately for both domains.

1.1.1.1 Levelset and distance functions

At any point $\mathbf{x}$ of the computational domain Ω, the levelset function α corresponds to the signed distance from Γ_{im}. In turn, the interface Γ_{im} is given by the zero isovalue of the function α:

$$\begin{aligned} \alpha(\mathbf{x}) &= \pm d(\mathbf{x}, \Gamma_{\text{im}}), \mathbf{x} \in \Omega, \\ \Gamma_{\text{im}} &= \{\mathbf{x}, \alpha(\mathbf{x}) = 0\}. \end{aligned} \tag{1.1}$$

We use the following sign convention: $\alpha \geq 0$ inside the solid domain defined by the interface Γ_{im} and $\alpha \leq 0$ outside this domain. Further details about the algorithm used to compute the distance will be given thereafter and the reader is invited to read [25] for an exhaustive overview of the method. It is also possible to use functions smoother than $d(\mathbf{x}, \Gamma_{\text{im}})$ away from Γ_{im} (see for example [26]).

As explained, the signed distance function is used to localize the interface of the immersed structure but it is also used to initialize the desirable properties on both sides of the latter. Indeed, for the elements crossed by the level-set functions, fluid-solid mixtures are used to determine the element effective properties. A Heaviside function $H(\alpha)$ is then defined as follows:

$$H(\alpha) = \begin{cases} 1 & \text{if } \alpha > 0 \\ 0 & \text{if } \alpha < 0 \end{cases} \tag{1.2}$$

The Heaviside function can be smoothed to obtain a better continuity at the interface [27] using the following expression:

$$H_\varepsilon(\alpha) = \begin{cases} 1 & \text{if } \alpha > \varepsilon \\ \frac{1}{2}\left(1 + \frac{\alpha}{\varepsilon} + \frac{1}{\pi}\sin\left(\frac{\pi\alpha}{\varepsilon}\right)\right) & \text{if } |\alpha| \leq \varepsilon \\ 0 & \text{if } \alpha < -\varepsilon \end{cases} \tag{1.3}$$

where ε is a small parameter such that $\varepsilon = O(h_{\text{im}})$, known as the interface thickness, and h_{im} is the mesh size in the normal direction to the interface. In the vicinity of the interface, it can be computed using the following expression:

$$h_{\text{im}} = \max_{j,l \in K} \nabla\alpha \cdot \mathbf{x}^{jl}, \tag{1.4}$$

where $\mathbf{x}^{jl} = \mathbf{x}^l - \mathbf{x}^j$ and K is the mesh element under consideration. According to the chosen approximations, the Heaviside function is then approximated using linear interpolations ($P1$) between fluid and solid properties or a piecewise constant interpolation ($P0$).

1.1.1.2 Levelset and anisotropic mesh adaptation

We combine next the levelset representation with an anisotropic mesh adaptation algorithm to ensure an accurate capturing of the discontinuities at the fluid-solid interface.

The levelset function intersects the mesh element arbitrarily. It is possible then to overtake the discontinuity appearing at the interface by using anisotropic mesh adaptation and regularization. The regularization parameter can be seen as the thickness or the resolution of the interface. It is shown that using local adaptivity, stretched elements at the interface are obtained which enables the resolution of the thickness to be very small and leads to very sharp interfaces, favorable for simulating fluid-structure interactions and conjugate heat transfer.

This anisotropic adaptation is performed by constructing a metric map that allows the mesh size to be imposed in the direction of the distance function gradients. We introduce first a metric which is a symmetric positive defined tensor that modifies the distance computation [28–31], such that:

$$||\mathbf{x}||_{\mathrm{M}} = \sqrt{{}^t\mathbf{x} \cdot \mathbb{M} \cdot \mathbf{x}}\ , \quad < \mathbf{x}, \mathbf{y} >_{\mathrm{M}} = {}^t\mathbf{x} \cdot \mathbb{M} \cdot \mathbf{y}\ . \tag{1.5}$$

In our context, the metric $\mathbb{M}$ can be regarded as a tensor whose eigenvalues are related to the mesh sizes, and whose eigenvectors define the directions for which these sizes are applied. For instance, using the identity tensor, one recovers the usual distances and directions of the Euclidean space. In our case, the direction of mesh refinement is given by the unit normal to the interface which corresponds to the gradient of the level set function: $\mathbf{x} = \boldsymbol{\nabla}\alpha/||\boldsymbol{\nabla}\alpha||$. A default mesh size, or background mesh size, h_d is imposed far from the interface and it is reduced as the interface comes closer. A likely choice for the mesh size evolution is the following:

$$h = \begin{cases} h_d & \text{if } |\alpha(\mathbf{x})| > \varepsilon/2 \\ \dfrac{2h_d(m-1)}{m\ \varepsilon}|\alpha(\mathbf{x})| + \dfrac{h_d}{m} & \text{if } |\alpha(\mathbf{x})| \leq \varepsilon/2 \end{cases} \tag{1.6}$$

Eventually, at the interface, the mesh size is reduced by a factor m with respect to the default value h_d. Then this size increases gradually till equalling h_d for a distance that corresponds to the half of a given thickness ε.

The unit normal to the interface $\mathbf{x}$ and the mesh size h defined above, lead to the following metric:

$$\mathbb{M} = C\,(\mathbf{x}\otimes\mathbf{x}) + \frac{1}{h_d^2}\,\mathbb{I} \qquad \text{with} \qquad C = \begin{cases} 0 & \text{if } |\alpha(\mathbf{x})| \geq \varepsilon/2 \\ \dfrac{1}{h^2} - \dfrac{1}{h_d^2} & \text{if } |\alpha(\mathbf{x})| < \varepsilon/2 \end{cases} \tag{1.7}$$

where $\mathbb{I}$ is the identity tensor. This metric corresponds to an isotropic metric far from the interface (with a mesh size equal to h_d for all directions) and to an anisotropic metric near the interface (with a mesh size equal to h in the $\mathbf{x}$ direction and equal to h_d in the other directions).

In practice, the mesh is generated in several steps using the MTC mesher developed by T. Coupez [32, 33] and through the CimLib library. This mesher is based on a topological optimization technique available in [31] for the anisotropic case. At each step of the refinement process, the mesh size converges locally toward the target size. This *a priori* method is simple and efficient. However, it has one disadvantage; the difficulty to control the total number of nodes, in particular for 3D industrial applications. Therefore we propose in this work to use the method developed in [34] and improved in [35]. The method is based on an a posteriori error estimator which allows more flexibility to adapt the mesh. Moreover with such a method it is possible to adapt the mesh not only on the level-set, but also on physical fields like the velocity or the temperature, and all at the same time.

For illustration, Figure 1.2 presents the zero isovalues of the levelset function for an immersed F1 car (left) and a helicopter (right). It clearly emphasizes the extremely stretched elements along the interfaces whereas the rest of the domain keeps the same background mesh size.

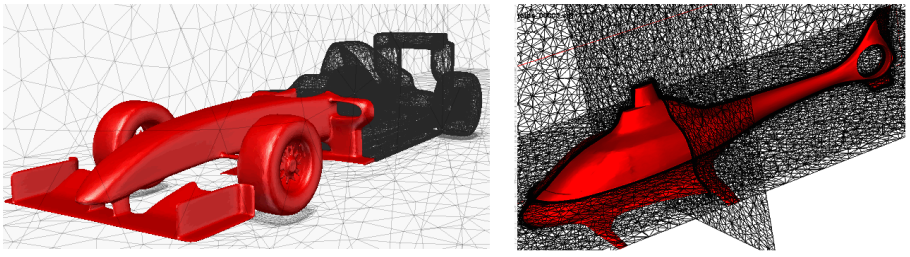


FIGURE 1.2: Anisotropic mesh adaptation at the fluid-solid interfaces

1.1.1.3 Levelset and mixing laws

Once the mesh is well adapted at the interface, the material distribution among the physical domains can be described by means of the levelset function. Consequently, the same set of equations; momentum equations, energy equation, the turbulent kinetic and dissipation energy equations, and the radiative transport equations are simultaneously solved over the entire domain with variable material properties. The use of the smoothed Heaviside function defined in (1.3) regularizes and enables the assignment of the right

properties on each side of the interface. The material properties such as density, initial temperature, dynamic viscosity, heat capacity and mean absorption coefficient, are computed as followed:

$$\begin{aligned}
\rho &= \rho_f H(\alpha) + \rho_s(1 - H(\alpha)) \\
\mu &= \mu_f H(\alpha) + \mu_s(1 - H(\alpha)) \\
\rho C_p &= (\rho_f C_{pf} H(\alpha) + \rho_s C_{ps}(1 - H(\alpha))) \\
\rho C_p T &= \rho_f C_{pf} T_f H(\alpha) + \rho_s C_{ps} T_s (1 - H(\alpha)) \\
\kappa &= \kappa_f H(\alpha) + \kappa_s(1 - H(\alpha))
\end{aligned} \tag{1.8}$$

However, as far as the thermal conductivity is concerned, linear interpolation would lead to inaccurate results. According to [36], one has to resort to the following law to ensure the conservation of the heat flux:

$$\lambda = \left(\frac{H(\alpha)}{\lambda_f} + \frac{1 - H(\alpha)}{\lambda_s} \right)^{-1} \tag{1.9}$$

1.2 Immersed NURBS

However immersed methods operate on non-body fitted grids which requires a special interface treatment. Indeed recent developments are focusing on issues related to the immersion of a surface mesh [25], the detection and the intersection algorithms for the interface and finally the transmission of boundary conditions between the solid and the fluid regions. In particular these methods appear to be limited by the quality and the accuracy of the surface mesh description of a given immersed solid.

We contribute a new approach for the immersion technique simplifying and bypassing the generation of a surface mesh. It is based on the use of Non Uniform Rational B-Splines (NURBS) curves or surfaces. These functions are used in the Computer Aided Design (CAD) field to represent simple or complex geometries. We compute the distance function from any point in the fluid mesh to these NURBS. Therefore instead of relying on the resolution of the surface mesh of the object, the proposed method uses directly the CAD definition and keeps the quality of its analytical description. In practice, it eliminates the surface mesh generation step and reduces the complexity to set up a Fluid-Structure application. Combined with anisotropic mesh adaption it provides an attractive immersed framework.

Linking the CAD and numerical simulation field has already been done before [37]. The aim of isogeometric analysis is to use an exact representation of the complex geometries and to replace the basis functions of the standard finite element method by the

ones of the NURBS. Therefore the Immersed NURBS Method combines the advantages of isogeometric analysis and the immersed methods. It capitalises on the exact geometry provided by the NURBS curves and surfaces recovered by the anisotropic mesh adaptation as well as on the flexibibility of the immersed methods.

The computation of the distance mainly relies on two steps: (i) finding a good initial guess [38] in order to (ii) use an iterative method to find the closest point [39]. Although, many methods and techniques have been already developed to compute the distance to NURBS functions, none of them has been used to compute level-set functions for immersed objects needed to solve FSI problems.

This work is organized as follows: in the first part (Part I) we introduce the NURBS functions as well as the theoretical methods that are essential to immerse NURBS based objects (Chapter 2). Then we present the new developped NURBS immersion method and test its rapidity and its accuracy (Chapter 3). Finally the method is coupled to anisotropic mesh adapation and is used to solve CFD applications (Chapter 4). In the second part (Part II), we introduce the used finite element solvers and justify the need of stabilization schemes (Chapter 5). Then we present the anisotropic mesh adaptation method (Chapter 6). We explain the concept and validate the method through academic cases. Afterwards we show the efficiency of the coupling between the stabilized solvers, the immersed volume method and the anisoptropic mesh adapation through complex industrial applications (Chapter 7).

Résumé français

Les problèmes d'interaction fluide-structure (IFS) sont présents dans de nombreux domaines tels que l'ingénierie mécanique et civile ou encore la biomécanique. Malgré l'amélioration permanente des technologies informatiques et des codes de calculs en mécanique des fluides, des problèmes persistent et empêchent la simulation numérique de manière précise des problèmes IFS.

Il existe deux approches pour simuler numériquement des problèmes IFS: les méthodes partitionnées et les méthodes monolithiques. Les méthodes partitionnées consistent à traiter les fluides et les solides de manière séparer et coupler les codes de calcul par des conditions aux limites. L'inconvénient majeur de telles méthodes réside dans le couplage des codes de calcul. Des instabilités peuvent également avoir lieu lorsque les ratio des propriétés physiques et fluides sont élevés. Les méthodes monolithiques quant à elles ne prennent en compte qu'un seul domaine de calcul incluant les fluides et les solides. Les solides sont immergés dans la partie fluide. Ainsi un seul code de calcul est nécessaire. L'intérêt de telles méthodes est notamment de réduire la complexité de génération de maillages. Les frontières du maillage n'ont donc pas à coïncider avec les interfaces fluide-solide. Par conséquent ces méthodes sont beaucoup plus flexibles lorsqu'il s'agit de changer la configuration de la simulation, par exemple rajouter un solide dans la simulation ou encore mettre un solide en mouvement.

La méthode d'immersion de volume est une méthode monolitique. Les solides sont immergés et repérés dans la partie fluide par une fonction distance signée. Ensuite les propriétés fluides et solides sont affectées par une fonction Heaviside lissée. Le logiciel Thost est basé sur la méthode d'immersion de volume. Ce logiciel d'aérothermie permet de simuler des procédés industriels tels que la trempe ou le chauffage dans des fours. L'utilisation de la méthode d'immersion de volume permet de s'affranchir des caractérisations de coéfficient d'échange, rendant la simulation numérique des procédés plus accessible. En effet ces coefficients d'échange peuvent s'avérer difficiles à quantifier. Le logiciel a montré de bonnes capacités à simuler avec précision des procédés complexes. Cependant les maillages fins nécessités par la complexité des problèmes engendrent des temps de calcul longs et rendent donc difficile l'utilisation du logiciel dans un contexte industriel.

Le but de cette thèse est donc d'améliorer les temps de calcul afin d'arriver à des simulations plus réalistes en terme d'exploitation. Pour cela nous présentons dans une première partie une modification de la méthode d'immersion de volume. Les géométries immergées dans les calculs sont généralement des maillages surfaciques. Nous proposons une méthode innovante pour immerger directement les objets à partir de leur fichiers CAO (Conception Assistée par Ordinateur), renforçant ainsi la fexibilité et la phylosophie de la méthode d'immersion. Dans la seconde partie nous présentons les méthodes numériques utilisées pour réaliser la simulation des procédés industriels. La méthode

d'immersion de volume est couplée à une adaptation dynamique de maillage anisotrope et des méthodes éléments finits stabilisées.

Chapter 2

Basics of NURBS

In this chapter we introduce the NURBS functions as well as fondamental tools needed for immersing CAD objects. In section 2.1 we remind the reader basic definitions, operations and concepts of NURBS. In section 2.2 we also introduce the key point of the Immersed NURBS Method which mainly consists in computing the distance from a point to a NURBS curve or surface. This problem has already been treated in the litterature, therefore we present attractive methods to compute the distance relatively to NURBS curves or surfaces.

2.1 Introduction to NURBS and general algorithms

A NURBS or Non-Uniform Rational B-Spline is a piecewise-polynomial parametric function. These functions were introduced in the 1950s [40, 41] in the industrial engineering field to represent complicated curved surfaces like ship hulls and aerospace exterior surfaces. They are now widely used in the Computer Aided Design (CAD) field and are the base of many designing softwares (CATIA, Pro Engineer, SolidWorks...). With such mathematical functions, it is possible to represent any geometry of different level of complexity. Their main advantage is that they can be locally modified by just moving control points without affecting the rest of the geometry. Figures 2.1 and 2.2 show examples of a NURBS curve and a NURBS surface with their corresponding control points.

2.1.1 Definition of NURBS curves and surfaces

The definition of a NURBS curve C is as follows:

$$C(u) = \frac{\sum_{i=0}^{n} N_{i,p}(u)\omega_i P_i}{\sum_{i=0}^{n} N_{i,p}(u)\omega_i} \tag{2.1}$$

where p is the degree of the curve, $N_{i,p}$ the basis functions, P_i the control points, $A = n+1$ the number of control points, ω_i the weights and u the parameter taking its values in the knot vector U. The knot vector U has $A+p+1$ knots and the first knot and the last knot are of multiplicity $p+1$ ($U = \{\underbrace{u_0, \ldots, u_0}_{p+1}, u_1, \ldots, u_{n-1}, \underbrace{u_n, \ldots, u_n}_{p+1}\}$). Therefore the number of nodes is directly linked to the degree of the curve and the number of control points. The multiplicity of the first and last nodes are also linked to the degree of the curve. The multiplicity of a node is the number of times it appears in the knot vector. The basis functions are defined by the Cox-De Boor recursion formula [42, 43]:

$$N_{i,0}(u) = \begin{cases} 1 & if\ u_i \leq u < u_{i+1} \\ 0 & otherwise \end{cases} \tag{2.2}$$

$$N_{i,p}(u) = \frac{u - u_i}{u_{i+p} - u_i} N_{i,p-1}(u) + \frac{u_{i+p+1} - u}{u_{i+p+1} - u_{i+1}} N_{i+1,p-1}(u),\ with\ p \in \mathbb{N}^* \tag{2.3}$$

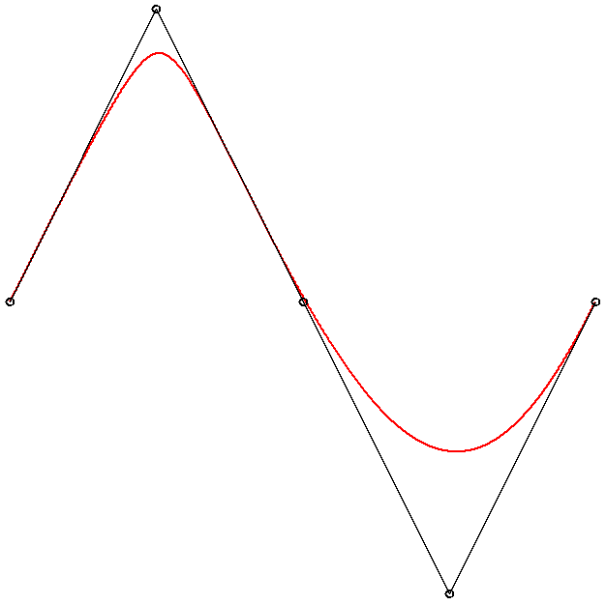


FIGURE 2.1: Example of a NURBS curve (red) and its control points (black circles) forming its convex hull (black lines)

The example given in Figure 2.1 shows a NURBS curve of order 4 (the order of a p-degree NURBS is $p+1$), with its 5 control points. The knot vector and the weights are the following : $U = \{0, 0, 0, 0, 0.5, 1, 1, 1, 1\}$, $W = \{1, 4, 1, 1, 1\}$. The set of segments binding the control points is called the control polygon. An important property of NURBS is that the curve (surface) lies inside the control polygon (this is called the convex hull property). Figure 2.1 shows that the curve is stretched by the control points having a weight of 4 (upper control point).

Following the definition given by (2.1), a NURBS surface S is simply the tensor product of two NURBS curves and can be defined as follows:

$$S(u,v) = \frac{\sum_{i=0}^{m}\sum_{j=0}^{n} N_{i,p}(u)N_{j,q}(v)\omega_{ij}P_{ij}}{\sum_{i=0}^{m}\sum_{j=0}^{n} N_{i,p}(u)N_{j,q}(v)\omega_{ij}} \tag{2.4}$$

where p and q are the polynomial degrees in the u and v directions, $N_{i,p}$ and $N_{j,q}$ the basis functions in the u and v directions, P_{ij} the control points, $M = m+1$ and $N = n+1$ the numbers of control points in the u and v directions, ω_{ij} the weights and u and v the parameters taking their values in the U and V knot vectors. The latters are constructed in the same way as mentionned previously in the NURBS curve definition.

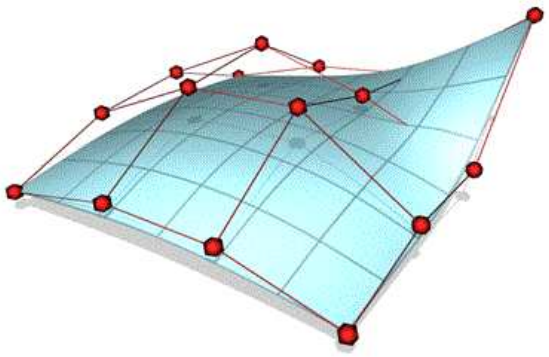


FIGURE 2.2: Example of a NURBS surface (blue) and its control points (red circles) forming its convex hull (red lines)

2.1.2 Definition of rational Bezier curves and surfaces

The definition of a rational Bezier curve D is as follows:

$$D(u) = \frac{\sum_{i=0}^{n} B_{i,n}(u)\omega_i P_i}{\sum_{i=0}^{n} B_{i,n}(u)\omega_i} \qquad 0 \leq u \leq 1 \tag{2.5}$$

where n is the degree of the curve, P_i the control points, $A = n+1$ the number of control points, ω_i the weights and $B_{i,n}$ the Bernstein polynomials defined by the following formula:

$$B_{i,n}(u) = \frac{n!}{i!(n-i)!}u^i(1-u)^{n-i} \tag{2.6}$$

The main difference between a rational Bezier and a NURBS curve or surface is therefore the basis functions. We can also notice that NURBS are piecewise polynomials. These

differences imply that rational Bezier curves usually need a higher degree than NURBS curves in order to fit complex shapes (a degree n is needed to fit $n+1$ data points). Moreover a change in control point will affect all the Bezier curve whereas the NURBS curve will only be modified locally. That is the reason why NURBS curves are more used in general.

Analogously to NURBS, we can define a rational Bezier surface as:

$$B(u,v) = \frac{\sum_{i=0}^{m}\sum_{j=0}^{n} B_{i,m}(u)B_{j,n}(v)\omega_{ij}P_{ij}}{\sum_{i=0}^{m}\sum_{j=0}^{n} B_{i,m}(u)B_{j,n}(u)\omega_{ij}} \qquad 0 \leq u,v \leq 1 \tag{2.7}$$

where m and n are the degrees of the surface in u and v directions, P_{ij} the control points, $M = m+1$ and $N = n+1$ the numbers of control points in the u and v directions, ω_{ij} the weights and $B_{i,m}$ and $B_{j,n}$ the Bernstein polynomials.

An important property of NURBS is the following: A NURBS curve (surface) that do not have any interior knot is a rational Bezier curve (surface) as the $N_{i,p}(u)$ reduce to the $B_{i,n}$.

2.1.3 Derivatives of NURBS

The kth derivative of the NURBS basis function $N_{i,p}$ with $p >= k$ is given by:

$$N_{i,p}^{(k)}(u) = \frac{p!}{(p-k)!}\sum_{j=0}^{k} a_{k,j}N_{i+j,p-k} \tag{2.8}$$

with

$$\begin{aligned}
a_{0,0} &= 1 \\
a_{k,0} &= \tfrac{a_{k-1,0}}{u_{i+p-k+1}-u_i} \\
a_{k,j} &= \tfrac{a_{k-1,j}-ak-1,j-1}{u_{i+p+j-k+1}-u_{i+j}} \quad j = 1,\cdots,k-1 \\
a_{k,k} &= \tfrac{-a_{k-1,k-1}}{u_{i+p+1}-u_{i+k}}
\end{aligned}$$

Expressing the definition of a NURBS curve differently:

$$C(u) = \frac{\sum_{i=0}^{n} N_{i,p}(u)\omega_i P_i}{\sum_{i=0}^{n} N_{i,p}(u)\omega_i} = \frac{A(u)}{w(u)}$$

the k^{th} derivative of a p-degree NURBS curve with $p >= k$ can be obtained by the following formula:

$$C^{(k)}(u) = \frac{A^k(u) - \sum_{i=0}^{k} \binom{k}{i} w^{(i)}(u) C^{(k-i)}(u)}{w(u)} \tag{2.9}$$

$$\text{with} \qquad A^{(k)}(u) = \sum_{i=0}^{n-k} N_{i,p-k}(u)\omega_i P_i^{(k)}$$

$$\text{and} \qquad P_i^{(k)} = \begin{cases} P_i & k = 0 \\ \frac{p-k+1}{u_{i+p+1}-u_{i+k}} \left(P_{i+1}^{(k-1)} - P_i^{(k-1)} \right) & k > 0 \end{cases}$$

Analogously, we can express the NURBS surface definition as follows:

$$S(u,v) = \frac{\sum_{i=0}^{m} \sum_{j=0}^{n} N_{i,p}(u) N_{j,q}(v) \omega_{ij} P_{ij}}{\sum_{i=1}^{m} \sum_{j=1}^{n} N_{i,p}(u) N_{j,q}(v) \omega_{ij}} = \frac{A(u,v)}{w(u,v)}$$

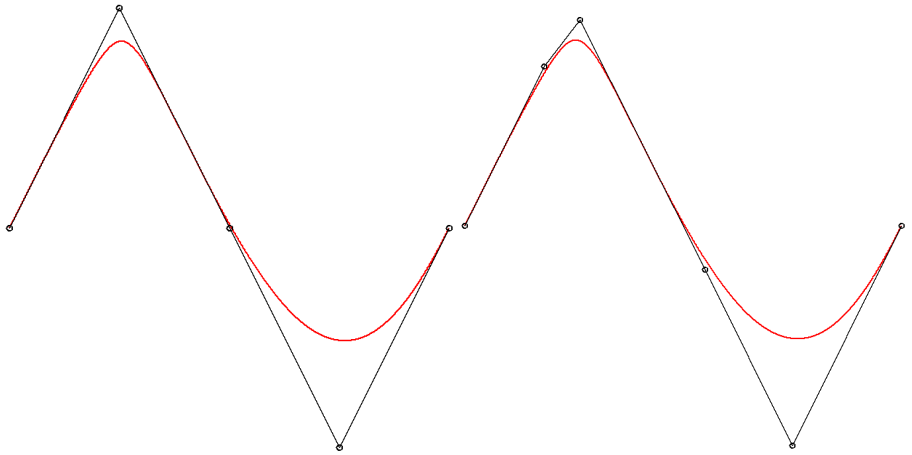
Then the k-l derivative of a NURBS surface of degree p and q with $p \geq k$ and $q \geq l$ can be obtained by the following formula:

$$S^{(k,l)} = \frac{1}{w} \left(A^{(k,l)} - \sum_{i=1}^{k} \binom{k}{i} w^{(i,0)} S^{(k-i,l)} - \sum_{j=1}^{l} \binom{l}{j} w^{(0,j)} S^{(k,l-j)} - \sum_{i=1}^{k} \binom{k}{i} \sum_{j=1}^{l} \binom{l}{j} w^{(i,j)} S^{(k-i,l-j)} \right) \tag{2.10}$$

$$\text{with} \qquad A^{(k,l)} = \sum_{i=0}^{m} \sum_{j=0}^{n} N_{i,p}^{(k)} N_{j,q}^{(l)} P_{i,j}$$

2.1.4 Knot insertion and NURBS subdivision

The knot insertion is a key tool of NURBS [39]. It consists in inserting the knot value $\overline{u}$ into the knot vector U of a NURBS curve C, with $u_0 \leq \overline{u} \leq u_n$ (Figure 2.3). The knot value $\overline{u}$ can be inserted multiple times until its multiplicity reaches the order of the curve ($m(u) = p + 1$). This process is the base of NURBS subdivisions. Once the knot value has a multiplicity equal to the order of the curve, the curve can been splitted into two sub-curves at this knot value $\overline{u}$. The same process can be applied to NURBS surfaces, giving four new sub-surfaces. Thus, coupling this process with the property



(A) (B)

FIGURE 2.3: NURBS curve (a) as presented previously in Figure 2.1 and curve (b) with its new control points (the knot value 0.2 has been inserted once)

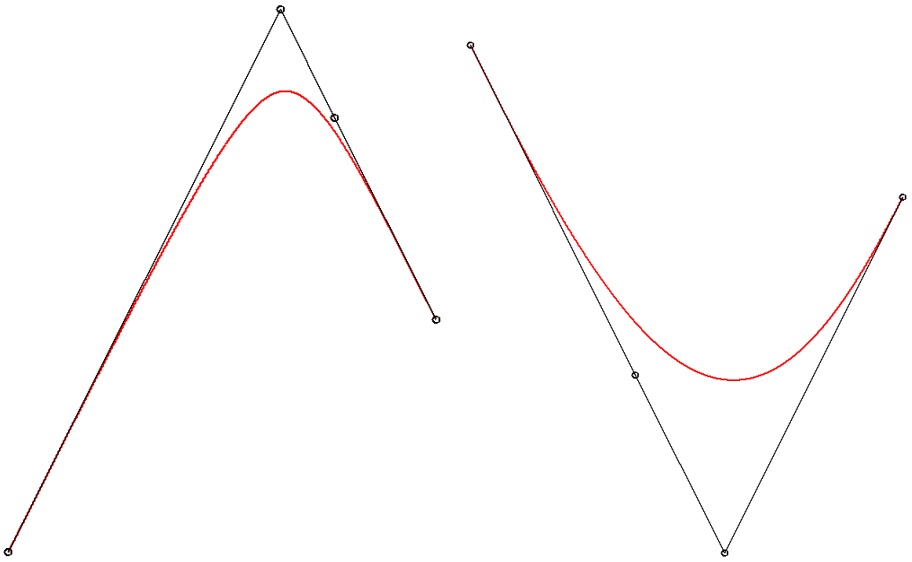


FIGURE 2.4: The two Bezier sub-curves obtained from the NURBS presented in Figure 2.1 by knot insertion (knot 0.5 has been inserted three times)

given into section 2.1.2, it is possible to subdivide a NURBS curve (surface) into a set of rational Bezier curves (surfaces) (Figure 2.4).

Inserting a knot has not only an impact on the knot vector U, it also modifies the control points (Figure 2.3). Indeed given a NURBS curve $C(u) = \frac{\sum_{i=0}^{n} N_{i,p}(u)\omega_i P_i}{\sum_{i=0}^{n} N_{i,p}(u)\omega_i}$, we can express it in its homogeneous form:

$$C^w(u) = \sum_{i=0}^{n} N_{i,p}(u) P_i^w, \qquad \text{with} \quad P_i^w = \omega_i P_i$$

We have seen that the new knot vector becomes $\overline{U} = u_0, ..., \overline{u}, ..., u_n$. The new expression of the curve is: $C^w(u) = \sum_{i=0}^{n+1} N_{i,p}(u) Q_i^w$. The new control points can be obtained as follows:

$$Q_i^w = \alpha_i P_i^w + (1 - \alpha_i) P_{i-1}^w$$

$$\text{where} \qquad \alpha_i = \begin{cases} 1 & i \leq k - p \\ \frac{\overline{u} - u_i}{u_{i+p} - u_i} & k - p + 1 \leq i \leq k \\ 0 & i \geq k + 1 \end{cases}$$

and k is the index of the greatest knot of U such that $u_k \leq \overline{u}$.

Inserting a knot into a NURBS surface is not far more complex. The same algorithm is used. The desired knot is inserted into the knot vector U or V of the surface S. Then suppose we want to insert knot $\overline{u}$ into U, thus a knot insertion is performed on the $n+1$ columns of the control points P_{ij}, resulting in a new control points sequence Q_{ij}, with $i \leq m + 2$ and $j \leq n + 1$. Analogously inserting $\overline{v}$ into V consists in a knot insertion on the $m + 1$ lines of the control points P_{ij}, resulting in a new control points sequence Q_{ij}, with $i \leq m + 1$ and $j \leq n + 2$.

2.1.5 Computing the product of two NURBS

Computing the product between two NURBS curves or surfaces is very useful, in particular for computing the distance between a point and a NURBS. In this case the self product of the NURBS is computed.

Given two NURBS curves c and d of degree p and q:

$$c(u) = \frac{\sum_{i=0}^{n_c} N_{i,p}(u) \omega_i^c P_i^c}{\sum_{i=0}^{n_c} N_{i,p}(u) \omega_i^c} \quad d(u) = \frac{\sum_{i=0}^{n_d} N_{i,q}(u) \omega_i^d P_i^d}{\sum_{i=0}^{n_d} N_{i,q}(u) \omega_i^d}$$

the target is compute the dot product: $h(u) = c(u).d(u) = \frac{\sum_{i=0}^{n_h} N_{i,p+q}(u) \omega_i^h P_i^h}{\sum_{i=0}^{n_h} N_{i,p+q}(u) \omega_i^h}$

The first step consists in computing the knot vector T of h. This can be done by rescaling the knot vectors R and S of c and d to the same parameter interval. Then each knot of R and S are copied into T with the following multiplicity:

$$m = \begin{cases} q + m_R & if \quad m_S = 0 \\ p + m_S & if \quad m_R = 0 \\ \max(q + m_R, p + m_S) & otherwise \end{cases}$$

where m_R and m_S are the multiplicities of the corresponding knot in R and S. The second step decomposes c and d into a set of rational bezier curves. Then the products between the Bezier curves is performed [44]:

$$P_k^{h,b} = \sum_{l=max(0,k-q)}^{min(p,k)} \frac{\binom{p}{l}\binom{q}{k-l}}{\binom{p+q}{k}} P_l^{c,b} P_{k-l}^{d,b} \qquad k = 0, \ldots, p+q$$

where $P_k^{h,b}$, $P_k^{c,b}$ and $P_k^{d,b}$ are the k^{th} control point of Bezier curve b. Finally the rational Bezier curves are recomposed to form the desired NURBS function h.

A similar technique can be extended to compute the dot product Z between two NURBS surfaces X and Y of degree p_X, q_X and p_Y, q_Y. The approach is the same than for NURBS curves and the coefficients are computed as follows:

$$P_{k,l}^{Z,b} = \sum_{i=max(0,k-p_Y)}^{min(p_X,k)} \sum_{j=max(0,l-q_Y)}^{min(q_X,l)} \frac{\binom{p_X}{i}\binom{p_Y}{k-i}\binom{q_X}{j}\binom{q_Y}{l-j}}{\binom{p_X+p_Y}{k}\binom{q_X+q_Y}{l}} P_{i,j}^{X,b} P_{k-i,l-j}^{Y,b}$$

with $\quad k = 0, \ldots, p_X + p_Y$ and $\quad l = 0, \ldots, q_X + q_Y$

A more detailed algorithm for the product of NURBS curves or surfaces can be found in [45].

2.2 Computing the distance to NURBS: a survey

Computing the minimum distance to NURBS is a complex problem that has already been treated and is still under research. It is used mainly for robotics, computer vision and geometric modeling, especially for selecting curves (surfaces), curve (surface) fitting problems or reconstructing curves (surfaces). This problem be can found in the litterature but has not been treated extensively as its complexity would justify.

Consider a NURBS curve C and a point P both lying in $\Re^a$, $a \in \{1; 2; 3\}$. Then the minimum distance problem consists in finding the point P_p on C such that $\|\overrightarrow{PP_p}\|$ minimizes the distance d between P and C. In other words, find the parameter u^* such that $d = \|C(u^*) - P\| = \min_{u \in U}(\|C(u) - P\|)$, where U is the knot vector of C. Now from a simple geometric analysis we can state that the projected point P_p that corresponds to the parameter u^* on the curve C must satisfy the following equation:

$$f(u) = (C(u^*) - P) \, .C^{'}(u^*) = 0 \tag{2.11}$$

where $C^{'}(u)$ is the first derivative of C. This equation means that the closest point to P on the curve C is the orthogonal projection of P on C (Figure 2.5a). Finding the minimum distance to C is therefore reduced to solving equation (2.11). The solution of equation (2.11) can be computed by using an iterative numerical method. All the complexity of the problem resides here. First, special cases have to be treated for which equation (2.11) is not satisfied. This occurs when the closest point is an extremity of the curve (Figure 2.5b). Second, depending on the starting guess value of the iterative algorithm, the returned closest point may have multiple possible values (Figure 2.5c). Thus finding the minimum distance between a point and NURBS has two main steps:

1. find a good initial guess on the curve
2. find the solution with an iterative method.

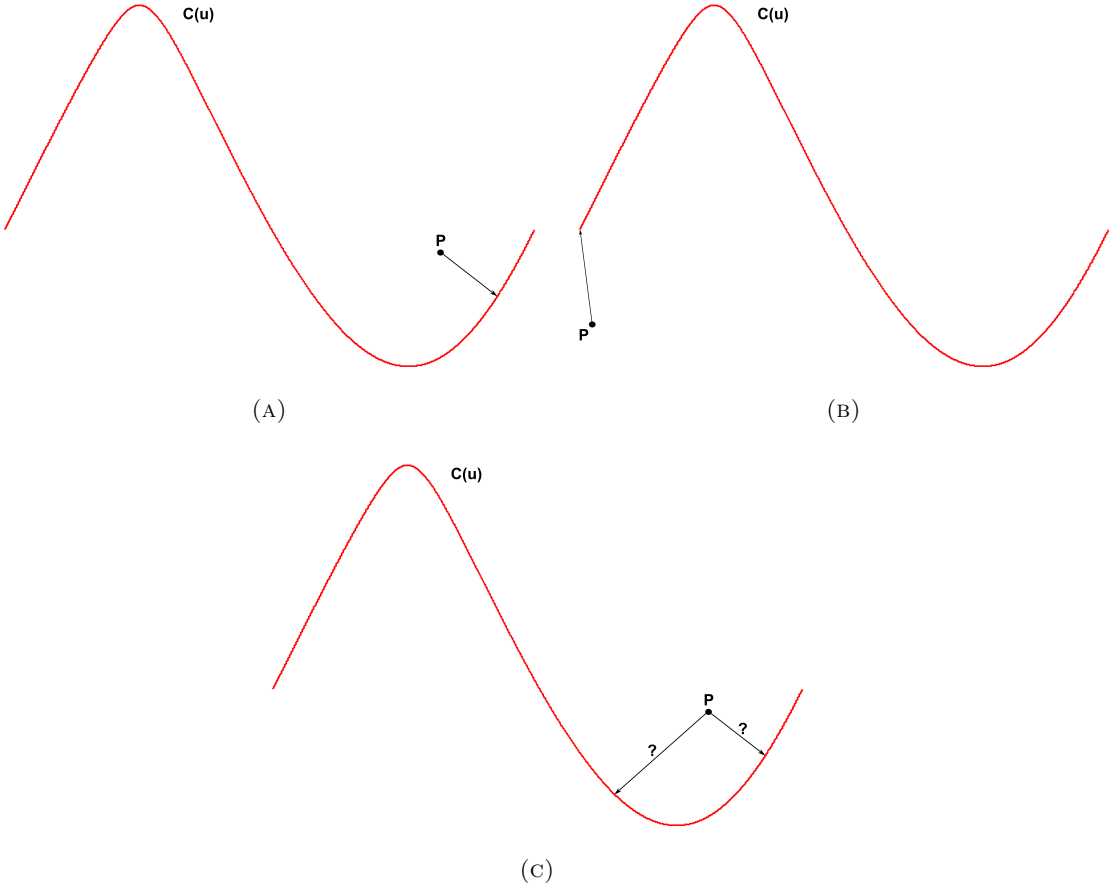


FIGURE 2.5: Different cases for the point projection problem. (a) The orthogonal projection of point P on C is the closest closest point. (b) The closest point is an extremity of the curve. (c) example of multiple solutions for orthogonal projection

2.2.1 Finding a good initial guess

Here we present attractive methods from the litterature used to find a good starting point for the iterative numerical method. This point is crucial for finding the closest point to a NURBS. Starting with an inappropriate guess value has two dramatic drawbacks. First, the iterative algorithm will do a lot of iterations, resulting in a high computational cost. Given the fact that we want to compute the distance between all the nodes of the computational mesh and the NURBS (possibly millions of nodes), a slight difference in the number of iterations can have a strong impact on the time spent to solve the distance function. Second, an inappropriate starting value will lead to an undesired solution (Figure 2.5c).

A method proposed in [39] consists in sampling n points on the NURBS and then choose the closest one to point P as a starting point for the iterative method. It is also possible to decompose the NURBS curve into a set of rational Bezier curves as a preparation phase [46]. Then the closest extremity of all the Bezier curves is taken as a first rough approximation of the distance. Finally the Bezier curves are subdivided again until they become flat enough and a check is performed on which one might lie the searched point. The solution is searched only on these remaining Bezier curves.

This method has been modified in [47]. The authors also decompose the NURBS curve into a set of rational Bezier segments. Then they check whether the control polygons of the Bezier curves are valid or not (a valid control polygon has no crossing edge and is convex). If a control polygon is not valid they subdivide it until it becomes valid. Next they test if the Bezier curves are flat enough and apply a scalar product criteria between the test point P and all the Bezier curves to eliminate the unnecessary ones. As a final stage they take the nearest point of the candidate points lying on the remaining Bezier curves as an initial guess for the iterative solving. But in [48] a counterexample on this method is given.

The solution can also be localized by subdividing recursively the NURBS curve [49]. After each split one of the new sub-curve is eliminated by applying a scalar product criteria. The splitting is done until a flatness condition is satisfied.

In [50] a novel approach is proposed to determine whether or not a NURBS sub-curve owns a unique solution. The authors first compute the squared distance function $(C(u) - P)^2$. Then they use an elimination circle and the squared distance function to eliminate the unnecessary curve parts. The squared distance function is very useful to determine if the solution is unique on a parameter interval. The uniqueness of the solution can be statued if the squared distance function has a u-shape, thus the minimum of the function is a global minimum. This function is computed by the algorithm described in section 2.1.5. This clipping technique is revisited in [51]. The elimination circle is replaced by a square and a coupling is done with the criterion given in [49].

Finally, all these methods aim at eliminating the parts or the parameter intervals of the curve that do not contain the closest point. Most of them are adaptable to NURBS surfaces.

2.2.2 Iterative methods to solve the point-distance to a NURBS

In this section we present iterative methods to find the closest point P to a NURBS curve C. We do not pay attention to the initial guess value but only to the method for solving equation 2.11. All these methods are general methods for finding the root of an equation and can be generalized for finding the closest point to any parametrized curve.

2.2.2.1 First order method

The first order method is a geometric iteration method. An exhaustive presentation of the method can be found in [52]. We just give here the guidelines of the algorithm. It uses the first derivative of the curve to compute the starting point for the next iteration. Here we are looking for the closest point of P on C, and the intial guess value is $C(u_0)$. We want to find the next parameter value u_1 of C so that $C(u_1)$ is closer to P than $C(u_0)$. Projecting point P onto the tangent line of C at parameter value u_0 gives a point Q that can be expressed as follows:

$$Q = C(u_0) + \Delta u.C^{'}(u_0) + o\left(\Delta u^2\right)$$

Therefore the method is linear and an expression of Δu can be found, thus we can compute the next parameter value u_1 (Figure 2.6). The outline of the algorithm is as follows:

First order method for parametric curves

---

1. compute the first derivative of C at parameter $u = u_0$
2. project point P onto the line $C^{'}(u_0)$ to obtain point Q
3. compute $\Delta u = \frac{\left\langle C^{'}(u_0), Q - C(u_0) \right\rangle}{\left\langle C^{'}(u_0), C^{'}(u_0) \right\rangle}$ [52]
4. update $u_1 = u_0 + \Delta u$
5. iterate until Δu becomes lower than a given tolerance or until the angle $\widehat{(C^{'}(u_0), Q - P)}$ is close enough to 90°.

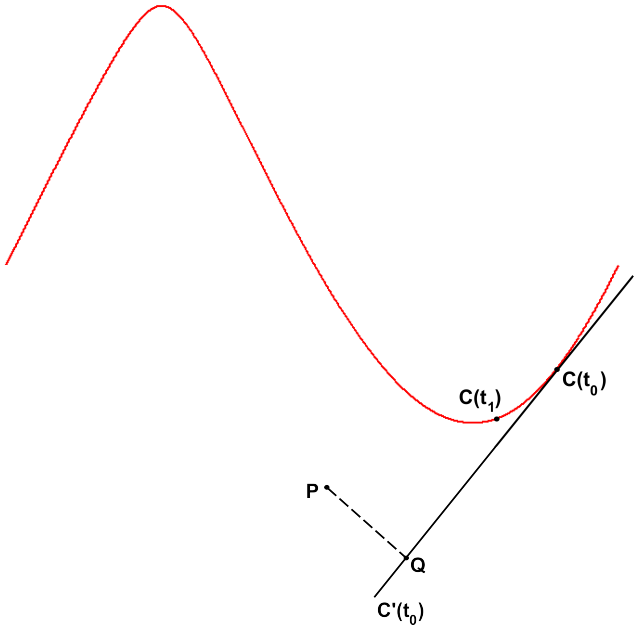


FIGURE 2.6: Outline of the first order method for projecting a point P on a curve C

The first order method is adaptable for computing the closest point to a surface S. Given the initial parameters u_0 and v_0, point P is projected onto the plane formed by $S_{,u}(u_0, v_0)$ and $S_{,v}(u_0, v_0)$, where $S_{,u}$ and $S_{,v}$ are the first derivatives of S in the u and v directions. Thus Q can be expressed as follows:

$$Q = S(u_0, v_0) + \Delta u S_{,u}(u_0, v_0) + \Delta v S_{,v}(u_0, v_0)$$

Multiplying by $S_{,u}(u_0, v_0)$ and $S_{,v}(u_0, v_0)$ we get:

$$\begin{cases} \langle S_{,u}(u_0, v_0), S_{,u}(u_0, v_0) \rangle \Delta u + \langle S_{,v}(u_0, v_0), S_{,u}(u_0, v_0) \Delta v \rangle = \langle Q - S(u_0, v_0), S_{,u}(u_0, v_0) \rangle \\ \langle S_{,u}(u_0, v_0), S_{,v}(u_0, v_0) \rangle \Delta u + \langle S_{,v}(u_0, v_0), S_{,v}(u_0, v_0) \Delta v \rangle = \langle Q - S(u_0, v_0), S_{,v}(u_0, v_0) \rangle \end{cases} \tag{2.12}$$

Solving this linear system of equation gives Δu and Δv and thus yields to the new values u_1 ans v_1. The process can be iterated similarly as the curve algorithm. Therefore the ouline of the alogrithm is the following:

First order method for parametric surfaces

---

1. compute the first derivatives in u and v directions of S at parameter values $u = u_0$ and $v = v_0$
2. project point P onto the plane formed by the two vectors $S_{,u}(u_0, v_0)$ and $S_{,v}(u_0, v_0)$ to obtain point Q [52]
3. solve the system (2.12) to obtain Δu and Δv
4. update parameter values $u_1 = u_0 + \Delta u$ and $v_1 = v_0 + \Delta v$
5. iterate until Δu and Δv become lower than a given tolerance or until both the angles $(\widehat{S_{,u}(u_0, v_0), Q - P})$ and $(\widehat{S_{,v}(u_0, v_0), Q - P})$ are close enough to 90°.

2.2.2.2 Second order method

The second order method is obtained using an analogous analysis. This method is more detailed in [52]. We just give here the guidelines of the algorithm. Instead of using a tangent line of C at parameter $C(u_0)$, we compute its curvature circle. The curvature circle has radius $\frac{1}{\kappa}$, with κ the curvature of the circle. The circle is on the side of the curve where $C''(u_0)$ points to. The radius if the circle is equal to:

$$r = \frac{\|C'(u_0)\|^3}{\det\left(C'(u_0), C''(u_0)\right)}$$

Point P is projected onto the curvature circle, giving point Q that can be expressed as follows:

$$Q = C(u_0) + \Delta u.C'(u_0) + \frac{\Delta u^2}{2}.C''(u_0) + o\left(\Delta u^3\right)$$

Therefore the method is quadratic and we can find an expression of Δu and compute the next parameter value u_1 (Figure 2.7). The ouline of the algorithm is as follows:

Second order method for parametric curves

---

1. compute the first and second derivatives of C at parameter $u = u_0$
2. compute the radius and the center of the curvature circle
3. project point P onto the circle to obtain point Q
4. compute $\Delta u = \frac{r}{\|C'(u_0)\|} \det\left(Q - C(u_0), C''(u_0)\right)$ [52]

5. update $u_1 = u_0 + \Delta u$
6. iterate until Δu becomes lower than a given tolerance or until the angle $(C'(u_0), \widehat{Q - P})$ is close enough to 90°.

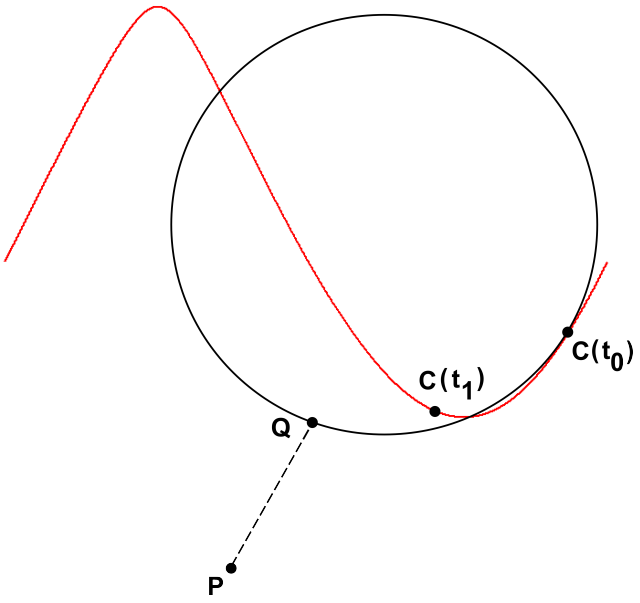


FIGURE 2.7: Outline of the second order method for projecting a point P on a curve C

The second order method is adaptable for computing the closest point to a parametric surface S. The following operators need to be defined in order to present the method:

$$n = \frac{S_u(u_0, v_0) \wedge S_v(u_0, v_0)}{\|S_u(u_0, v_0) \wedge S_v(u_0, v_0)\|}, \qquad g_{ij} = \langle S_i, S_j \rangle, \qquad h_{ij} = \langle n, S_{ij} \rangle$$

The point P can be expressed as follows:

$$P = S(u_0, v_0) + \lambda_u S_{,u}(u_0, v_0) + \lambda_v S_{,v}(u_0, v_0) + \nu n$$

Multiplying by $S_{,u}(u_0, v_0)$ and $S_{,v}(u_0, v_0)$, we get:

$$\begin{cases} \langle S_{,u}(u_0, v_0), S_{,u}(u_0, v_0) \rangle \lambda_u + \langle S_{,v}(u_0, v_0), S_{,u}(u_0, v_0) \lambda_v \rangle = \langle Q - S(u_0, v_0), S_{,u}(u_0, v_0) \rangle \\ \langle S_{,u}(u_0, v_0), S_{,v}(u_0, v_0) \rangle \lambda_u + \langle S_{,v}(u_0, v_0), S_{,v}(u_0, v_0) \lambda_v \rangle = \langle Q - S(u_0, v_0), S_{,v}(u_0, v_0) \rangle \end{cases}$$

Therefore, λ_u and λ_v can be computed as solution of this regular system of linear equations. Then the normal curvature at point $S(u_0, v_0)$ with normal vector $\lambda_u S_u(u_0, v_0) + \lambda_v S_v(u_0, v_0)$ is given by:

$$\kappa = \frac{h_{uu}\lambda_u^2 + 2h_{uv}\lambda_u\lambda_v + h_{vv}\lambda_v^2}{g_{uu}\lambda_u^2 + 2g_{uv}\lambda_u\lambda_v + g_{vv}\lambda_v^2}$$

The curvature circle at point $S(u_0, v_0)$ has radius $\frac{1}{\kappa}$ and its center is $S(u_0, v_0)+rn$. Thus we can project point P onto the curvature circle to obtain point Q. Then the new guess parameter values are computed with point Q. The outline of the algorithm is as follows:

Second order method for parametric surfaces

---

1. compute the first derivatives of S in the u and v directions at parameter values $u = u_0$ and $v = v_0$
2. compute n, λ_u and λ_v
3. compute the curvature κ
4. project point P onto the curvature circle to obtain point Q
5. compute $t = \sqrt{2r\frac{\det(\lambda_u S_u(u_0,v_0)+\lambda_v S_v(u_0,v_0), Q-S(u_0,v_0))}{\|\lambda_u S_u(u_0,v_0)+\lambda_v S_v(u_0,v_0)\|}}$
6. update $u_1 = u_0 + \Delta u$ and $v_1 = v_0 + \Delta v$ with $\Delta u = t\lambda_u$ and $\Delta v = t\lambda_v$
7. iterate until $\Delta u \Delta v$ becomes lower than a given tolerance or until both the angles $(\widehat{S_u(u_0, v_0), Q - P})$ and $(\widehat{S_v(u_0, v_0), Q - P})$ are close enough to 90°.

2.2.2.3 Newton-Raphson method

The well-known Newton-Raphson method is famous thanks to its rate of convergence which is quadratic. This method is well presented in [39, 53]. Its main drawback is the possible convergence to a local minimum or maximum, as already mentioned previously (Figure 2.5c), depending on the initial guess value. Geometrically, the method consists in finding the intersection of the tangent of the function at the guess value and the zero-value axis. The next parameter is the one corresponding to the intersection. The method is applied to find the closest point to a NURBS curve in [39]. The guidelines are the following.

Given the function $f(u) = (C(u) - P).C'(u)$, we try to find the root of this function. We start from the initial guess point $C(u_0)$. Thus the Newton-Raphson method gives:

$$u_1 = u_0 + \Delta u, \qquad \text{with} \quad \Delta u = -\frac{f(u_0)}{f'(u_0)} = -\frac{(C(u_0) - P).C'(u_0)}{(C'(u_0))^2 + (C(u_0) - P).C''(u_0)}$$

Therefore the outline of the algorithm is as follows:

Newton-Raphson method for parametric curves

---

1. compute $f(u_0)$ and $f^{'}(u_0)$
2. compute Δu and update $u_1 = u_0 + \Delta u$
3. iterate until $\Delta u \leq \epsilon$ or $f(u_i) \leq \epsilon$, with ϵ being a prescribed tolerance.

The method can be easily extended to compute the minimum distance between a point P and a NURBS surface S and the problem statement is the following [53]:

$$\begin{cases} a(u,v) = (S(u,v) - P).S_u(u,v) = 0 \\ b(u,v) = (S(u,v) - P).S_v(u,v) = 0 \end{cases} \tag{2.13}$$

The problem is transformed by solving iteratively the following system:

$$\begin{bmatrix} a_{,u}(u_i,v_i) & a_{,v}(u_i,v_i) \\ b_{,u}(u_i,v_i) & b_{,v}(u_i,v_i) \end{bmatrix} \begin{bmatrix} \Delta u \\ \Delta v \end{bmatrix} = \begin{bmatrix} -a(u_i,v_i) \\ -b(u_i,v_i) \end{bmatrix} \tag{2.14}$$

where a_u, a_v, b_u and b_v are the partial derivatives respectively in the u and v directions of a and b. Replacing (2.13) in (2.14) yields:

$$J_i. \begin{bmatrix} \Delta u \\ \Delta v \end{bmatrix} = \begin{bmatrix} -(S(u_i,v_i) - P).S_{,u}(u_i,v_i) \\ -(S(u_i,v_i) - P).S_{,v}(u_i,v_i) \end{bmatrix} \tag{2.15}$$

$$\text{with } J_i = \begin{bmatrix} \|S_{,u}(u_i,v_i)\|^2 + (S(u_i,v_i) - P).S_{,uu}(u_i,v_i) S_{,u}(u_i,v_i).S_{,v}(u_i,v_i) \\ +(S(u_i,v_i) - P).S_{,uv}(u_i,v_i) \\ \\ S_{,u}(u_i,v_i).S_{,v}(u_i,v_i) + (S(u_i,v_i) - P).S_{,vu}(u_i,v_i) \|S_{,v}(u_i,v_i)\|^2 \\ +(S(u_i,v_i) - P).S_{,vv}(u_i,v_i) \end{bmatrix} \tag{2.16}$$

Finally the parameters are computed by solving the following system:

$$\begin{bmatrix} u_{i+1} \\ v_{i+1} \end{bmatrix} = \begin{bmatrix} \Delta u \\ \Delta v \end{bmatrix} + \begin{bmatrix} u_i \\ v_i \end{bmatrix} \tag{2.17}$$

Thus the outline of the algorithm is the following:

Newton-Raphson method for parametric surfaces

---

1. compute $a(u_0, v_0)$, $b(u_0, v_0)$ and J_0
2. compute Δu and Δv by solving system (2.16)
3. update $u_1 = u_0 + \Delta u$ and $v_1 = v_0 + \Delta v$
4. iterate until u_1 and v_1 do not change significantly or both equations in (2.13) are satisfied under a given precision

2.2.2.4 Hybrid Newton-Raphson method

The main drawback of the Newton-Raphson method is its poor global convergence property. The method may converge to a local extremum depending on the starting value. It is possible to improve this convergence property by combining the Newton-Raphson with the bisection method, therefore yielding to a hybrid method [54]. The algorithm uses the bisection method when Newton-Raphson find a solution out of bounds or when it does not converge rapidly enough.

The outline of the algorithm is the following:

Hybrid Newton-Raphson method for parametric curves

---

1. given a specified parameter interval $[a, b]$, with $f(a) < f(b)$, compute $f(a)$, $f(b)$, $f(u_i)$ and $f^{'}(u_i)$ (equation (2.11))
2. if $\left[(u_i - f(b)).f^{'}(u_i) - f(u_i)\right]\left[(u_i - (f(a), f(b)).f^{'}(u_i) - f(u_i)\right] > 0$ or $|2f(u_i)| > |\Delta u_{i-1}.f^{'}(u_i)$, go to step 3 (Newton-Raphson is out of range or not decreasing fast enough). Otherwise go to step 4
3. compute $\Delta u_i = \frac{1}{2}(f(b) - f(a), f(b)))$ and update $u_{i+1} = f(a) + \Delta u_i$. Go to step 5
4. compute $\Delta u_i = -\frac{f(u_i)}{f^{'}(u_i)}$ and update $u_{i+1} = u_i + \Delta u_i$. Go to step 5
5. update a or b, if $f(u_{i+1}) < 0$ $a = u_{i+1}$, otherwise $b = u_{i+1}$
6. iterate until Δu_i becomes lower than a prescribed precision.

2.2.2.5 Brent-Dekker method

Similarly to the hybrid Newton-Raphson method, the Brent-Dekker method uses a combination of several methods to improve and optimize the reliability and the rate of convergence [54]. It is a mix of the bisection and the secant methods and inverse quadratic interpolation. Brent's method capitalizes on the superlinear rates of convergence of the inverse quadratic interpolation and the secant method while keeping the sureness of the bisection method. Brent has prooved that the method converges as long as the given interval of the treated function contains a root. The outline of the method is the following:

Brent-Dekker method for parametric curves

---

1. given a specified parameter interval $[a, b]$, with $|f(a)| > |f(b)|$, compute $f(a)$, $f(b)$ and set $c = a$
2. if $f(a) \neq f(c)$ and $f(b) \neq f(c)$ then use inverse quadratic interpolation:
   $s = \frac{af(b)f(c)}{(f(a)-f(b))(f(a)-f(c))} + \frac{bf(a)f(c)}{(f(b)-f(a))(f(b)-f(c))} + \frac{cf(a)f(b)}{(f(c)-f(a))(f(c)-f(b))}$
   otherwise use secant method:
   $s = b - f(b)\frac{b-a}{f(b)-f(a)}$
3. if $s \notin \left[\frac{3a+b}{4}, b\right]$ or $|s-b| \geq \frac{|b-c|}{2}$ then use bisection method $s = \frac{a+b}{2}$
4. iterate until $f(b)$, $f(s)$ or $|b-a|$ becomes lower than a prescribed precision.

This method is very powerful for a one-dimensional root-finding when only the values of the function, and not the ones of its derivatives, are available. For example it can be used to find the minimum value of the squared distance function.

2.2.2.6 Biarc approximation method

The biarc approximation is an iterative geometric method analogous to the first and second order methods. It has been developed by [55]. Unlike Newton-Raphson and Brent methods, it is very specific to the point projection and inversion onto planar parametric curves. For each iteration, a local biarc is contructed at the guess point and point P is projected onto this biarc. The method takes advantage of the iterative algorithms mentioned above (first order, second order and Newton-Raphson) by correcting the iterative solutions thanks to the biarc approximation.

A biarc is constituted by two circular arcs, having the same length in that case. Its contruction can be done by simply defining boundary conditions, i.e its two ending points and the tangents at these two points. The outline of the biarc approximation method is as follows:

Biarc method for parametric curves

---

1. given u_0 compute u_1 using a step of first order, second order or Newton-Raphson method
2. compute $C(u_0)$, $C(u_1)$, $C^{'}(u_0)$ and $C^{'}(u_1)$
3. construct the biarc defined by $C(u_0)$, $C(u_1)$, $C^{'}(u_0)$ and $C^{'}(u_1)$
4. project point P onto the biarc to obtain point Q and its corresponding parameter s on the biarc
5. correct $\Delta u = u_1 - u_0$ by $\Delta u = s\Delta t$
6. iterate until Δu becomes lower than a given tolerance or until the angle $(\widehat{C^{'}(u_0), Q - P})$ is close enough to 90°.

2.3 Conclusion

In this chapter we have introduced the NURBS which are piecewise-polynomial parametric functions used in various domains, especially for geometric representations. Their main advantage is their ease of manipulation while possibly representing any geometry. The basic operations of NURBS are mainly knot insertion, patches subdivision and multiplication. All these operators are used in the litterature in order to compute the distance from a point to a NURBS curve or surface, and more precisely to find a good initial guess value to solve the distance problem. Otherwise the iterative method may converge to a local minimum instead of a global one if the problem has multiple solutions. Moreover this minimizes the number of iterations to find the solution and thus reduces the computationl cost. As the objective is to compute the distance between a point and NURBS curves or surfaces for all the nodes of a computational mesh (possibly millions of nodes), every computational time saving is crucial. Various iterative methods can be used to find the closest point to a curve or surface. The iterative methods presented above have been developed and the tests results are provided in the next chapter.

Résumé français

Dans ce chapitre sont présentées les fonctions NURBS. Ces fonctions mathématiques paramétriques et polynomiales par morceaux furent inventées dans les années 1950 [40, 41] pour représenter des formes complexes tels que les carènes de bateau ou les carosseries des voitures. Aujourd'hui elles sont la base de nombreux logiciels de conception assistée par ordinateur (CAO). Leur principal avantage est qu'on peut modifier localement leur forme en bougeant simplement un point de contrôle sans affecter le reste de la courbe ou de la surface.

Afin d'immerger des courbes ou surfaces NURBS dans les simulations numériques, il est nécessaire de calculer une fonction distance par rapport à ces courbes ou surfaces analytiques. Les outils de base permettant ce calcul tels que l'insertion de noeuds, la subdivision, la décomposition en courbes ou surfaces de Bézier rationnelles ou le produit de deux NURBS sont décrits [39] dans la section 2.1.

Le calcul de la distance entre un point et une courbe NURBS peut-être assimilé à un problème de résolution d'équation. Le but étant de trouver la racine de l'équation (2.11). Plusieurs méthodes sont accessibles dans la littérature pour résoudre ce problème. Elles consistent pour la plupart à résoudre l'équation (2.11) par une méthode itérative. Il est donc nécessaire dans un premier temps de trouver une valeur initiale judicieuse pour la méthode itérative. Si la valeur initiale est trop loin de la solution, la méthode itérative risque fortement de diverger ou de converger autour d'un minimum local si la solution est multiple (Figure 2.5). La majorité des auteurs propose donc de subdiviser la courbe NURBS en courbes rationnelles de Bézier et d'éliminer les sous-courbes ne contenant pas la solution. Ainsi on s'affranchit d'une possible convergence vers un minimum local et on minimise le nombre d'itérations de la méthode itérative. Sachant que le calcul de distance entre un point et une NURBS doit être effectué pour chaque noeud du maillage de calcul (potentiellement plusieurs millions), le moindre gain en temps de calcul est significatif. La méthode itérative la plus répendandue est celle de Newton-Raphson [39] mais d'autres méthodes tels que la méthode de Brent [54] ou celle de l'approximation par Biarc [55] sont également utilisées.

Chapter 3

Immersed NURBS Method

In this chapter we present the method developed to immerse CAD objects. Indeed, in CAD files, each object is commonly characterized by NURBS curves or surfaces. The standard file format used to deal with CAD is the IGES format, which stands for Initial Graphics Exchange Specification. This file format has been created in the 1980s to exchange graphics and geometry data. It is widely used by the community and is a standard format in all CAD softwares. The reader is invited to read more about IGES in [39, 56]. We use the GoTools library to read iges files and perform basic operations on NURBS. The GoTools library is developed by SINTEF, which is a research organisation in Scandinavia (http://www.sintef.no/Projectweb/Geometry-Toolkits/GoTools).

The first section of this chapter introduces the level-set function and explains different ways to compute this function (relatively to simple analytical objects or complex objects represented by a surface mesh). In section 2 the new Immersed NURBS Method is presented. Methods and results are shown to find a good initial guess for the distance iterative solving of the distance between a point and a NURBS. Different iterative methods presented in Chapter 2 are also developed and tested.

3.1 Level-set

3.1.1 Level-set and surface mesh

The level-set function was introduced in [57]. This function is used in many fields, like image processing, computer graphics, computational geometry, optimization, numerical modelling, etc... In the field of computational mechanics it is used to locate the interface between a fluid and a solid. Its target is to describe the interface between several bodies. The level-set function gives at any node of the computational mesh the minimum distance

to a structure. This distance takes a negative sign if the node is outside of the solid. Figure 3.1 illustrates the immersed volume method and shows the level-set of a circle.

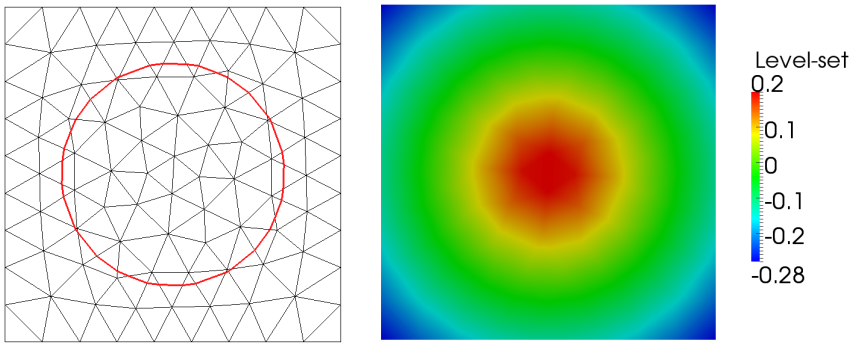


FIGURE 3.1: Immersion of a circle: the zero isovalue of the level-set in red and the mesh (left), and the level-set function (right)

Therefore the computation of the level-set consists in getting the distance relatively to the body (for example the circle) at every nodes of the mesh. Thus for analytical functions like the one presented in Figure 3.1, its computation is obvious and reads at any node:

$$\alpha\left(\vec{x}\right) = R - \sum_{i=1}^{d} \sqrt{x_i^2}$$

where α is the level-set, R the radius of the circle, d the dimension of the space and $\vec{x}$ the coordinates of the node.

On the other hand if the geometry of the immersed body is more complex we do not have the analytical function to compute the level-set (Figure 3.2). Thus the common way to immerse the body is to build its surface mesh, and then compute the distance relatively to this surface mesh. In fact for every node of the computational mesh, the distance between the node and each facet of the surface mesh is computed. Then the closest facet is selected and the sign of the distance is calculated depending on the normal of the facet. Obviously such an algorithm is time consuming because it has order of $O\left(n^2\right)$ time complexity. A faster algorithm is presented in [25]. The computational mesh is segmented in hierarchical boxes, and the facets of the surface mesh are stored in these boxes. Thus, depending on which box the node belongs to, only the distance to the facets situated in the box in question is computed. Therefore unnecessary operations are avoided. The order of time complexity of such an algorithm is $O\left(n \log n\right)$. This method has been developed in Thost, and we have chosen to use it when immersing surface meshes.

The computation of the distance between a node and a facet of a surface mesh (i.e. a triangle in 3D) mainly consists in finding in which area of the seven ones lies the node

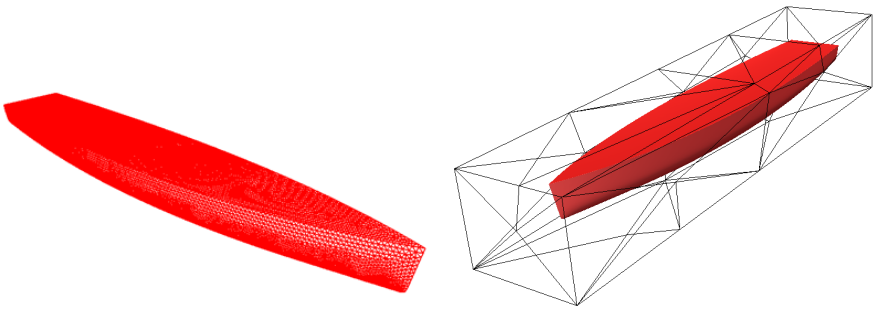


FIGURE 3.2: Immersion of a ship hull: its surface mesh (left), and the zero isovalue of the level-set in red and the computational mesh (right)

relatively to the triangle (Figure 3.3). The distance calculation depends on this area [53]. In addition to the distance, a quality parameter is also computed [25].This quality parameter compares the computed distance to the facet with the projected distance relatively to the normal of the facet. The smaller this difference, the better the quality. The segmentation into areas is useful, thereby the distance from a node to a triangle is not done by projecting the node onto the triangle normals, and thus the triangle vertices and edges can be the solution of the projection. Therefore the projection of the node is always on the triangle.

Other methods to compute the distance to a surface mesh are available in the litterature. An interesting survey is developed in [58]. An attractive method is presented in [59]. The principle is to start from the nodes of the computational mesh positioned at the interface of the immersed object. The distance of the nodes belonging to these elements is then computed and propagated to the other elements thanks to the Eikonal equation:

$$\left(\sum_{i=1}^{n} N_{,i}^2\right) d^2 + \left(\sum_{i=1}^{n} d_i N_{,i}\right) 2d + \left(\sum_{i=1}^{n} N_{,i}^2 - 1\right) = 0$$

with $d_i = \sum_{k=1}^{n} dn_k N_{k,i}$, d being the distance to be calculated at the node in question, $N_{,i}$ the derivative of the basis functions relatively to the i-direction, n the number of nodes of the element, dn_k the known distance at node k and $N_{k,i}$ the derivative of the basis function at node k relativeky to the i-direction. Thanks to this equation it is possible to find the distance from the node to the interface. Step by step the iteration is done over all the elements, thus the distance function propagates. Note that, as the method starts with the nodes positioned at the interface (i.e. having a distance equal to zero), it restricts to cases where the interface passes exactly through the element nodes.

Although immersing objects by their surface mesh (commonly .stl files) is generic, we emphasize that the precision of the level-set is highly dependent on the quality of the surface mesh. Obviously, a coarse surface mesh will imply a poor level-set precision compared to the initial CAD geometry of the object. More than that, after building the CAD of the object, generating the surface mesh requires time and specific knowledge.

Therefore immersing directly the CAD geometry would lead to a significant gain in the accuracy of the level-set as well as a more generic way to set up a physical problem, which strengthens the phylosophy of immersion techniques.

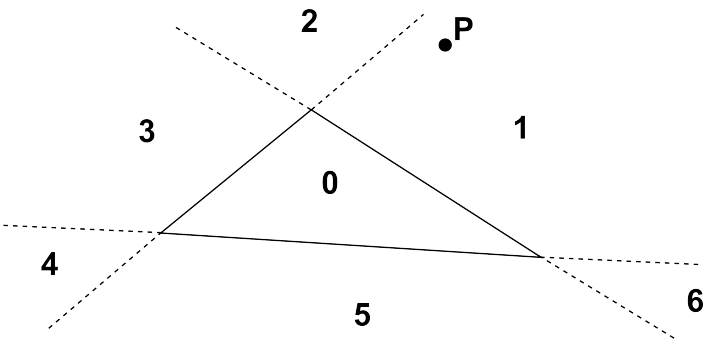


FIGURE 3.3: Scheme of the areas of a triangle in which a node P can lie

3.1.2 Level-set and NURBS

This section describes the new NURBS immersion technique. It computes the distance function from any point of the computational mesh to the NURBS to obtain the zero iso-value of this function. The computation of the distance mainly relies on patching the NURBS functions [38] and using a Newton method [39]. Although, many methods and techniques have been already developed to compute the distance to NURBS functions (see Chapter 2), none of them has been used to compute level-set functions for immersed objects needed to solve FSI problems. Computing the level-set to NURBS based objects consists in computing the distance function at each node of the computational mesh. The main steps are to find a good initial guess for the iterative method and then perform this iterative method to find the distance. Finaly the distance has to be signed.

3.1.2.1 The closest point problem

We have implemented and tested several of the methods described previously. All these methods can be found in the litterature, they are not new, but to our knowledge this is the very first time they are used to compute level-sets of immersed structures. To compute the level-set of an immersed object which is represented by NURBS functions, we need to compute the distance from every node of the computational mesh to the NURBS functions, and then sign it. Finding the distance from a point (i.e. a node in our case) to NURBS can be done by solving the closest point problem (Equation (2.11)), which in fact can be seen as a root finding problem [53].

This kind of equations can be solved using iterative methods like the ones presented in Chapter 2. It is obvious that the efficiency, the reliability and the computational cost of these methods are highly dependent on the initial guess value. The closer this

value to the solution is, the more stable and faster is the convergence. We recall that the distance is computed for every node of the computational mesh, which can have millions of nodes. Therefore finding a good initial guess value is crucial in order to save computational time. Different methods for finding a good initial guess are presented thereafter.

3.1.2.1.a Bezier patches decomposition

To ensure that Equation (2.11) has always a solution we first check if the extremities of the NURBS curve or surface do not include the closest point (Figure 3.4). Therefore we use the useful criterion given in [49]. This criterion based on simple scalar products defines an area where the extremities of the curve are the closest point (Figure 3.4). The criterion is defined as follows:

Algorithm 1 Extremity criterion

if $\forall$ i $\in$ [0,n] $\mathbf{P_0P_i}.\mathbf{PP_0} \geq 0$ **then**
  P_0 is the closest point
else if $\forall$ i $\in$ [0,n] $\mathbf{P_nP_i}.\mathbf{PP_n} \geq 0$ **then**
  P_n is the closest point
end if

P being the query point (node of the computational mesh) and P_i the control points of the NURBS curve. Therefore we start by checking if the closest point of the curve (respectively surface) is one of the extremities. If it is the case then the distance is computed automatically. Otherwise the point is on the inner part of the curve (respectively surface), and Equation (2.11) can be solved.

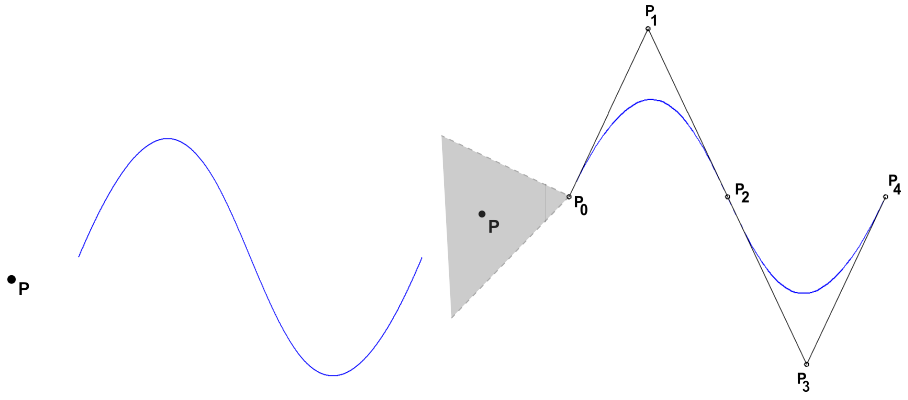


FIGURE 3.4: Case of a NURBS curve of order 4 (blue) where the closest point of P on the curve is one of the extremities (left). The grey zone (right) satisfies the extremity criteria, with P_0 the closest point, P_i being the control points of the curve.

Once the solution has been determined to be on the inner part of the function, we have to check if the solution is unique. Indeed Equation (2.11) can have multiple solutions, as shown on Figure 3.5. To avoid this issue we subdivide the function into rational Bezier

patches. We recall that rational Bezier patches are NURBS functions with no interior knots (Figure 3.6). The steps of the NURBS subdivision into rational Bezier patches are presented in Chapter 2.

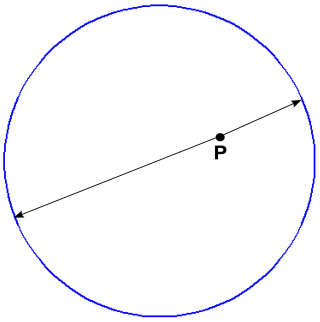


FIGURE 3.5: Case of a NURBS curve of order 3 (blue) where Equation (2.11) has multiple solutions (black arrows).

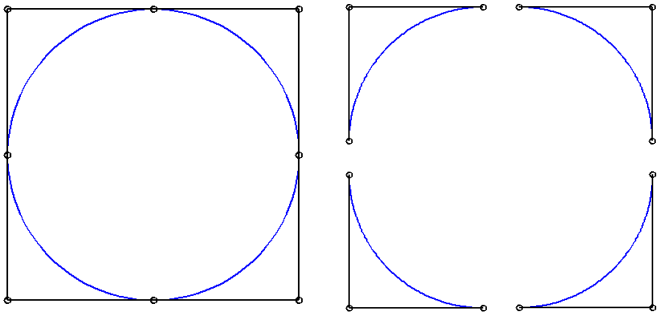


FIGURE 3.6: Example of a NURBS curve of order 3 (blue) and its control points (left) subdivided into four rational Bezier segments (right).

Once we have subdivided the function, we search the solution on each patch, and keep the smallest distance. The outline of the algorithm is depicted in Algorithm 2.

Algorithm 2 Closest point with Bezier patches decomposition

Check if the closest point is one of the extremities with Algorithm 1
if one of the extremities is the solution **then**
  Compute the distance
else
  Subdivide the function into rational Bezier patches
  Compute one distance with an iterative method for each patch (the starting value can be the middle of the patch)
  The minimum distance between all patches is the solution
end if

3.1.2.1.b Bezier patches elimination

A method to compute the distance to NURBS functions has been presented in the previous section. In fact, this method is not optimal as the iterative method is run for

each patch of the NURBS. Thus multiple iterations are done for each patch of a NURBS of an object, knowing that an object can have several NURBS. To avoid unnecessary computations we use again the method presented in [49].

This time the criterion is not only used to check if one of the extremities of the NURBS function is the closest point, but also to eliminate the Bezier patches that do not contain the solution. In fact if the solution is not an extremity of the function, it has to be on the inner part of the function. Thus it is also on the inner part of a Bezier patch, or on a cusp of the curve. Therefore we use the criterion on each Bezier patch to check if the closest point on the patch is one of its extremities. In the case the result is true, it implies that the solution is not on the inner part of this patch, and thus that the patch does not contain the solution (Figure 3.7). Therefore the patch is eliminated (Algortihm 3). The outline of the algorithm is presented in Algorithm 4.

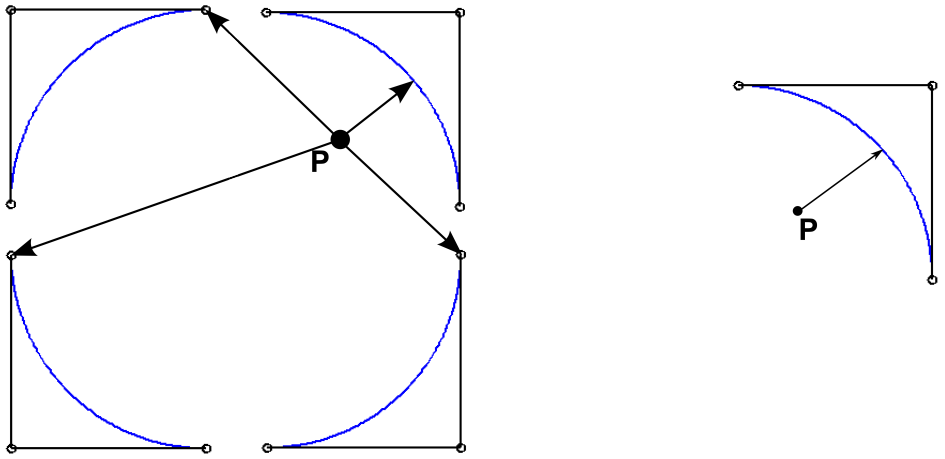


FIGURE 3.7: Example of the patches elimination of a NURBS curve of order 3 (blue). All the extremities of the patches are tested with Algorithm 1 (left). Only the patch which extremeties are not the closest point from P remains (right). The solution is searched only on this patch.

Algorithm 3 Extremity criterion for Bezier patches

if $\forall$ i $\in$ [0,n] $\mathbf{P_{k,0}P_{k,i}}.\mathbf{PP_{k,0}} \geq 0$ **then**
  $P_{k,0}$ is the closest point
else if $\forall$ i $\in$ [0,n] $\mathbf{P_{k,n}P_{k,i}}.\mathbf{PP_{k,n}} \geq 0$ **then**
  $P_{k,n}$ is the closest point
end if

3.1.2.1.c Bezier patches segmentation

The Methods to compute the distance to NURBS functions presented in the previous sections work for low order curves or surfaces but encounter issues as the order increases. Figure 3.8 shows a NURBS curve of order 4 before and after the subdivision step and the

Algorithm 4 Closest point with Bezier patches elimination

```
Check if the closest point is one of the extremities of the curve or surface with Algorithm 1
if one of the extremities is the solution then
  Compute the distance
else
  Subdivide the function into rational Bezier patches
  for each patch do
    Check if the closest point on the patch is one of the extremities of the patch with Algorithm 3
    if the closest point is not one of the extremities then
      Compute the distance to the patch with an iterative method
    else
      Eliminate the patch of the search list
    end if
  end for
  if all the patches have been eliminated then
    for all knots of the NURBS of multiplicity equal to the order do
      Compute the distance
    end for
  end if
  The solution is the minimum distance
end if
```

associated multiple solutions. Despite the subdivision of the NURBS into rational Bezier patches, both solutions remain on the same patch. Therefore, depending on the initial guess value, the result returned by the iterative method can be wrong. Subdividing NURBS into rational Bezier patches is not sufficient enough for computing the distance function.

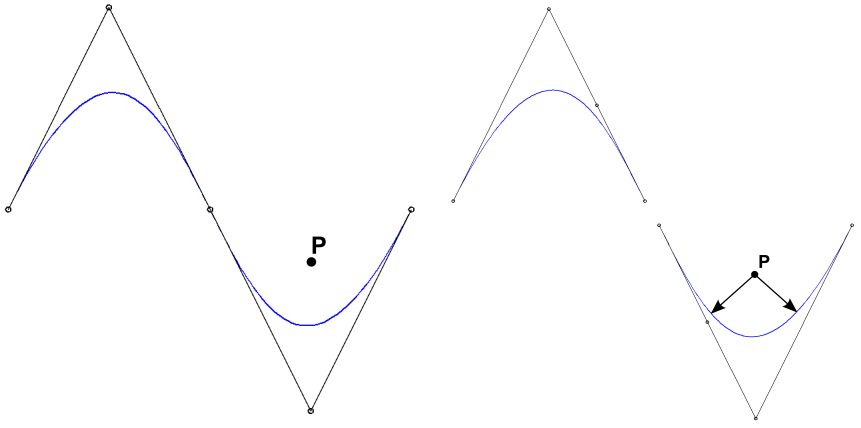


FIGURE 3.8: Example of a NURBS curve of order 4 (blue) and its control points (left) subdivided into two rational Bezier segments (right). Both solutions remain on the same patch.

To avoid such an issue, we apply the method presented in [49] which consists in subdividing the NURBS curve (respectively surface) iteratively into two subcurves (respectively

four subsurfaces). After each subdivision, we test the patches with the useful criterion also provided in [49] to eliminate those not containing the solution. The patches are subdivided iteratively until the control polygon of the patches reach a flatness condition [47]. In other words the control points of the patches are close enough to a straight line. The flatness condition ϵ can be the sum of the distances of the control points to the support curve of the patch:

$$\sum_{i=1}^{n-1} d(P_i, \mathbf{P_0P_n}) < \epsilon \tag{3.1}$$

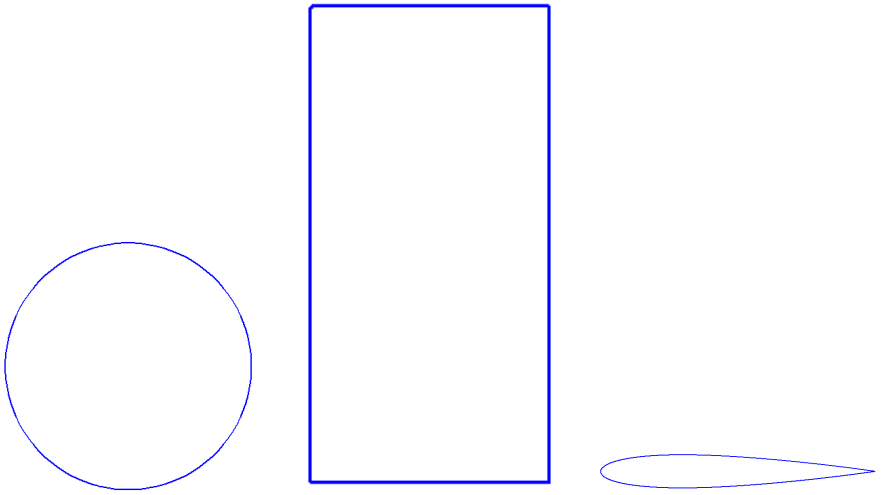
,where P_i is a control point of the patch and $d(P_i, \mathbf{P_0P_n})$ is the distance between P_i and the segment $\mathbf{P_0P_n}$ formed by points P_0 and P_n. The outline of the general algorithm is described in Algorithm 5.

Algorithm 5 Closest point with Bezier patches segmentation

```
Check if the closest point is one of the extremities of the curve or surface with Algorithm 1
if one of the extremities is the solution then
  Compute the distance
else
  Subdivide the function at the middle knot
  for each patch do
    Check if the closest point on the patch is one of the extremities of the patch with Algorithm 3
    if the closest point is one of the extremities then
      Eliminate the patch of the search list
    else
      Subdivide the patch, eliminate the initial one and add both new to the search list
    end if
  end for
  if the patch search list is empty then
    for all knots of the NURBS of multiplicity equal to the order do
      Compute the distance
    end for
  else
    for all remaining patches do
      Compute the distance to the patch with an iterative method
    end for
  end if
  The solution is the minimum distance
end if
```

In fact there is an optimal precision for the flatness condition. The smaller this value, the closer the initial guess to the solution. Because the shorter the Bezier patches, the faster the iterative algorithm but the slower the subdivision step. Therefore we have tested the algorithm for several flatness precision values. The results are presented in table 3.1. These computations have been done for NURBS geometries (Figure 3.9) immersed in a mesh made of 131 nodes. The iterative method used is Newton-Raphson. It shows that the smaller the flatness parameter, the slower the method. This means that the subdivision procedure has a significant computational cost compared to the search of the solution with the iterative method. In fact the subdivision procedure implies NURBS knot insertions (see Chapter 2) which is definitly time consuming. Figure 3.10 clearly

shows that as we increase the number of subdivisions of the curve, the percentage of time taken by the elimination step drastically increases. We notice that this value can reach more than 90% of the computational time.



(A) (B) (C)

FIGURE 3.9: NURBS circle (a), rectangle (b) and NACA0012 (c).

Geometry	ϵ	Computational time (s)
a	0.1	0.02819590
a	0.01	0.07502700
a	0.001	0.18242800
b	0.1	0.00830412
b	0.01	0.00834584
b	0.001	0.00827694
c	0.1	0.12500600
c	0.01	0.20554700
c	0.001	0.33502700

TABLE 3.1: Computational time in seconds of the distance calculation for different 2D geometries, and different flatness condition ϵ (Equation 3.1) for 131 points

3.1.2.1.d Squared distance method

Another method is presented in [50, 60]. This method is similar to the one used in [49], and thus from those presented in the previous sections. It is also based on the test of the uniqueness of the solution on a patch. In this method a different criterion is used. This criterion is based on the computation of the squared distance function. In fact for each patch a squared distance function can be computed. Depending on the shape of this function, the uniqueness of the solution can be determined. Figure 3.11 shows a NURBS curve and the point P from which we want to know the distance to the curve. The squared distance function of the curve c, $(c(u) - P)^2$, is also provided. Note that

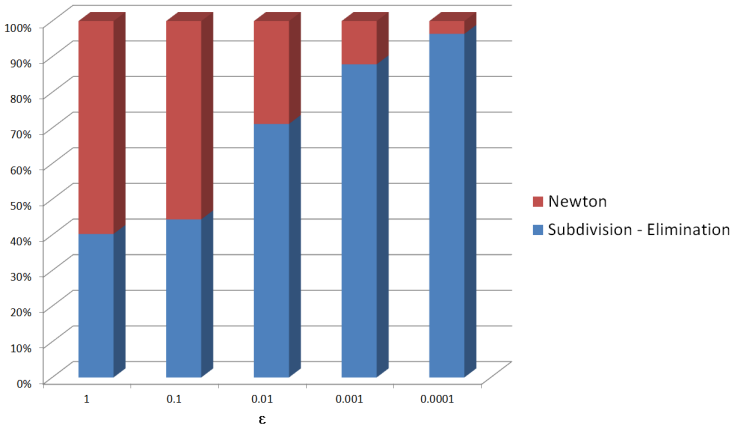


FIGURE 3.10: Percentage spent by the subdivision and the elimination step versus the Newton method among the total computational time of the distance relatively to geometry a for different flatness condition values.

the squared distance function is point dependant. This implies that from a numerical point of view, the squared distance function cannot be computed as a prephase unlike the Bezier patches subdivision. Especially when using mesh adaptation, the nodes move and thus the squared distance function changes as soon as a mesh adaptation is performed.

The method consists in testing the uniqueness of the solution on a patch with the squared distance function, and then subdividing the patch if the solution is multiple. In fact if the squared distance function has u-shape, then it means there is only one minimum, and that this minimum is global, whereas on the example shown in Figure 3.11 the squared distance function admits two local minimum values. The test of the a u-shape of the squared distance function can be done by verifying the constant decrease and increase of the control points values of the squared distance function. The outline of the algorithm is given in Algorithm 6.

3.1.2.1.e Sampling method

An alternative to all these methods using the subdivision technique is the one presented in [39]. This method is easier to implement. It consists in sampling a certain number of points on the NURBS curve or surface (equally or randomly spaced). Then the closest sample point is chosen as the initial guess of the iterative method (Figure 3.12). The outline of the method is detailed in Algorithm 7.

In fact, a balance has to be found in order to optimize the rate of convergence of the algorithm. The more sample points the closer the initial guess to the solution and thus the faster the iterative method, but the slower the search of the guess point. Therefore there is an optimal number of points to sample in order to have a optimized algorithm. We have tested the rate of convergence of the method for several numbers of sample points on three different geometries presented in Figure 3.13. The results are presented

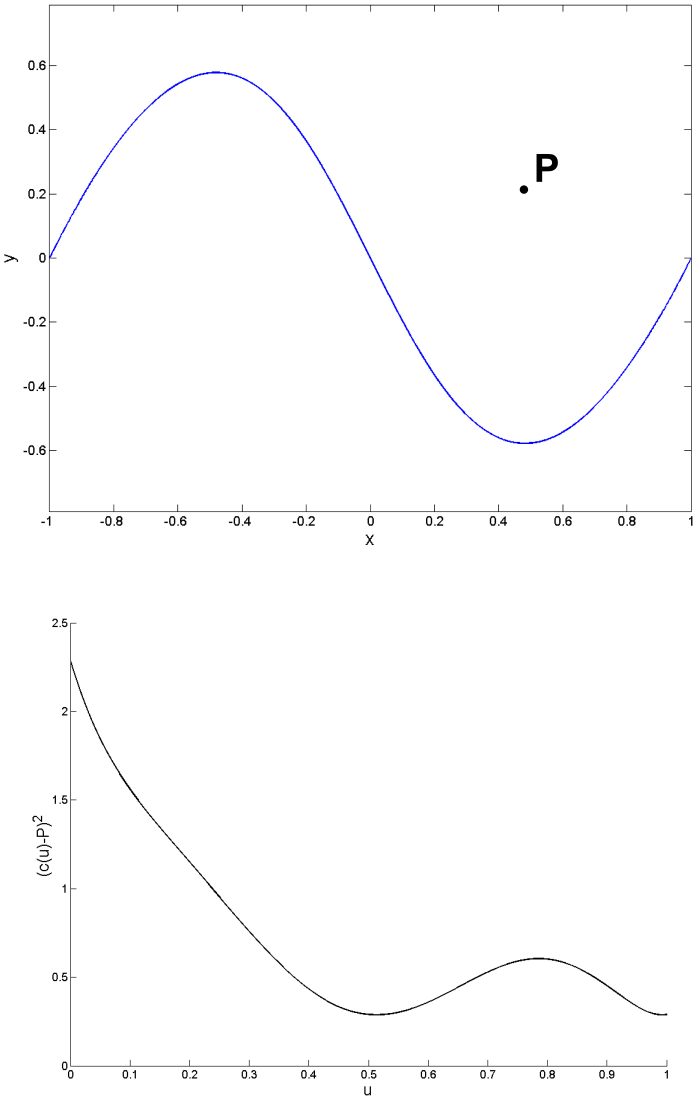


FIGURE 3.11: Example of NURBS curve c of order 4 in blue (top) and its squared distance function $(c(u) - P)^2$ (bottom).

in table 3.2. All these distance computations have been done using the Newton-Raphson iterative method. From the results we can see that the fastest computation times for the two NURBS curves are obtained when using a number of sample points between 50 and 100. We point out that with less than 20 sample points the method is not robust and provides wrong distance functions. The other information is that the optimal number of sample points for the NURBS surface is 1000 as the computational time is the fastest. Again, with less than 1000 sample points the method does not provide the right distance function. The results presented in table 3.2 have been obtained by sampling the points equally along the curves of surfaces. We have also tested to sample them randomly but have not noticed a significant change in the computational times. The repartition of the computational time with respect to the number of sample points for geometry a is

Algorithm 6 Closest point with squared distance function

Check if the closest point is one of the extremities with Algorithm 1
if one of the extremities is the solution **then**
  Compute the distance
else
  Subdivide the function at the middle knot
  for each patch **do**
    Translate the patch from a vector P
    Compute the squared distance function of the patch $(c_k(u) - P)^2$ with the method presented in Chapter 2
    if the squared distance function has not a u-shape **then**
      Subdivide the patch, eliminate the initial one and add both new to the search list
    end if
  end for
  if the patch search list is empty **then**
    for all knots of the NURBS of multiplicity equal to the order **do**
      Compute the distance
    end for
  else
    for all remaining patches **do**
      Compute the distance to the patch with an iterative method
    end for
  end if
  The solution is the minimum distance
end if

Algorithm 7 Closest point with sampling method

Sample a number of points on the curve or surface
Find the closest sample point $\tilde{P}$ to the query point P
Find the distance solution with an iterative method by using $\tilde{P}$ as the initial guess

provided in Figure 3.14.

3.1.2.2 Comparison of the selecting methods

As the computational time is a key point of this work, all the methods presented in this section to select the best initial first guess have been developed and tested (Algorithms 2, 4, 5, 6 and 7). We remind that these methods will be used to immerse each object of the computation every time the mesh, and thus the level-set change. Therefore it is essential to optimize the level-set computation by using the fastest and more robust method.

We have tested all the methods on different NURBS based geometries presented in Figure 3.15. The target is to check the robustness and the rate of convergence of the methods for objects formed by one or several NURBS curves or surfaces.

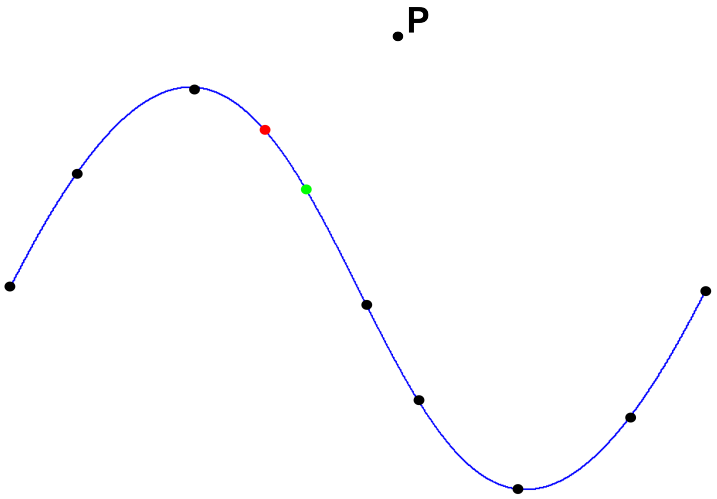
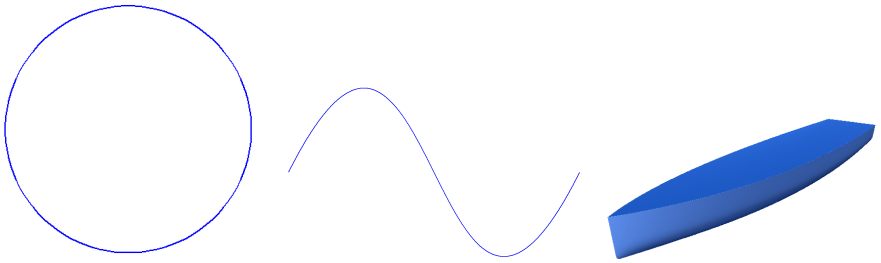


FIGURE 3.12: Example of a NURBS curve c of order 4 in blue, a sample of points in black, the closest point among all the sample points in green, which is thus the initial guess, and the solution of the closest point on the curve in red.



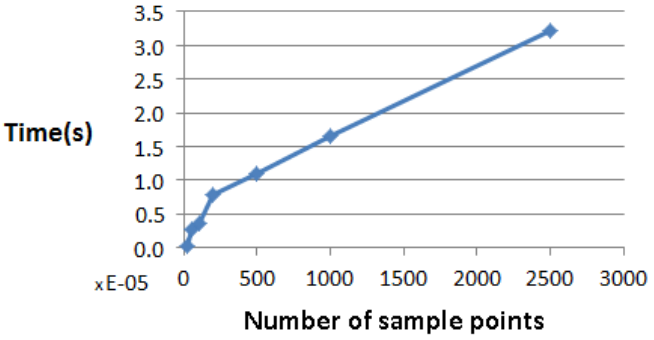
(A) (B) (C)

FIGURE 3.13: NURBS circle (a), curve of order 4 (b) and ship hull (c).

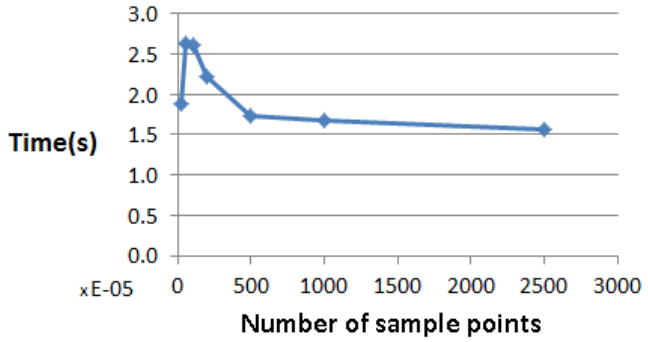
# sample points	Geometry a	Geometry b	Geometry c
20	0.00196004	0.00188398	-
50	0.00192809	0.00182009	-
100	0.00195503	0.00179291	-
200	0.0020659	0.00187588	-
500	0.00219607	0.00217986	-
1000	0.00247812	0.0024209	77.4026
1750	-	-	77.9146
2500	0.00384283	0.00379419	80.9115
5000	-	-	91.4979
10000	0.00986409	0.00977707	106.77
25000	-	-	198.618
100000	0.0830529	0.0836859	738.31

TABLE 3.2: Computational time in seconds of the distance computation for different 2D geometries, and different number of sample points. The mesh used for gemetries a and b contain each 100 nodes, and 102040 nodes for geometry c.

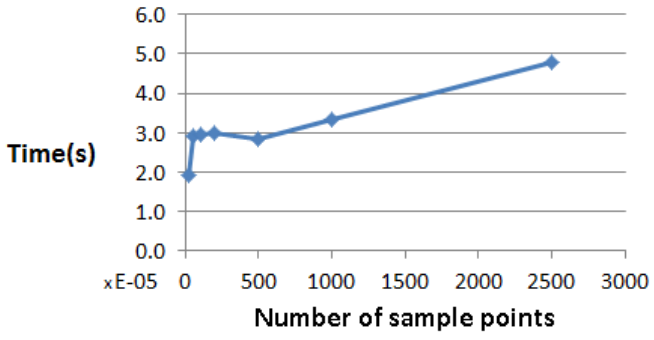
The results presented in Table 3.3 show that the most competitive method is the sampling method. In fact it is faster to search the closest point among a sample of points



(A)



(B)



(C)

FIGURE 3.14: Average time spent per node for the closest sample point search step (a), the Newton method (b) and average total time (c).

than to do NURBS operations like knot insertion, subdivision. Especially the Squared distance method is far slower than the other methods as the computation of the self product of NURBS is a costly operation [51]. As the sampling and the Bezier patches elemination methods are the most efficient methods, we challenge one to each other for the computation of the distance relatively to a NURBS 3D object (Figure 3.13 (c)). The interest here is also to compute the distance to an object made of several NURBS

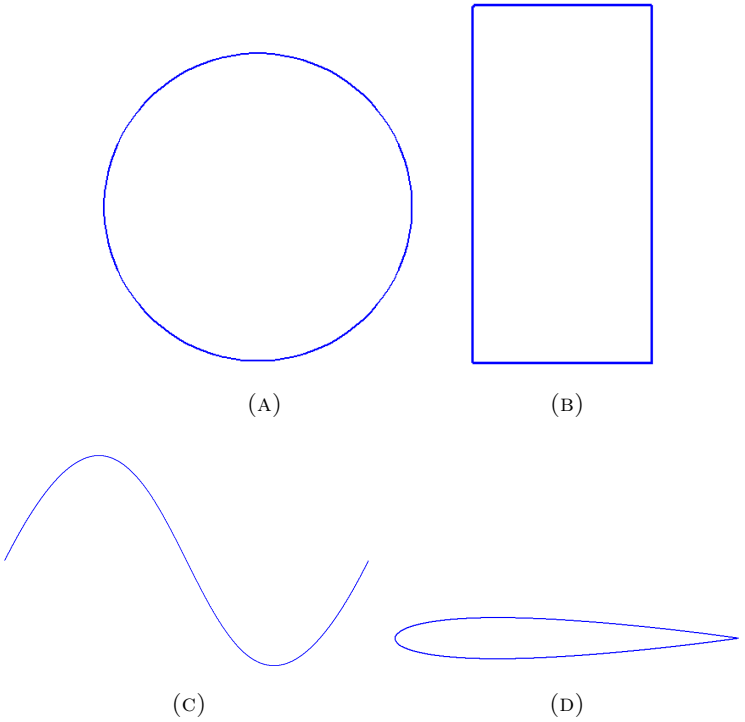


FIGURE 3.15: NURBS circle (a), rectangle (b), curve of order 4 (c) and NACA0012 profile (d).

Method	Geometry a	Geometry b	Geometry c	Geometry d
Bezier Patches Decomposition	0.00558008	0.004492282	0.005278581	0.046601527
Bezier Patches Elimination	0.00484445	0.002088435	0.004887575	0.032048931
Bezier Patches Segmentation	0.02152358	0.006339023	0.021700833	0.095424427
Squared Distance	0.66955400	0.284027176	0.666665230	4.526911546
Sampling	0.00195503	0.000835374	0.00179291	0.013353721

TABLE 3.3: Computational time in seconds of the distance computation for different 2D geometries. The meshes used contain each 100 nodes.

surfaces, four in this case. The mesh used to compute the distance to the object contains $102,040$ nodes. The results of Table 3.4 confirm that the sampling method is faster, even for NURBS surfaces. It also demonstrates that both methods are well parallelized.

# cores	Sampling method	Bezier Patches Elimination method
1	77.40	160.36
2	39.47	82.19
4	24.76	51.59
8	12.64	26.18

TABLE 3.4: Computational time in seconds of the distance computation for a geometry of Figure 3.13 (c). The mesh used for the geometry contains $102,040$ nodes. The computational time is provided for different numbers of cores

3.1.2.3 Iterative methods

The first step for computing the distance relatively to a NURBS based object is to find a good starting value for the iterative method that will be used to compute the distance. We have implemented and presented a list of selecting methods and have shown that the best choice is to sample a number of reasonnable points on the NURBS curve (or surface) and use the closest one to the query point (a node of the computational mesh in our case) as a starting value of the iterative method. In the previous subsection only the Newton-Raphson method has been used to compute the distance. In this section, different iterative methods will be used, comparisons will be made and conclusions will be drawn. All these methods have been implemented and tested with the geometries *a*, *b*, *c* and *d* shown in Figure 3.15. The tests have been done only on 2D geometries, and only the best method has been implemented in 3D. The results are shown in table 3.5. Recall that the details and the algorithms related to these methods were given in Chapter 2.

Geometry	Brent	Hybrid Newton-Raphson	Newton-Raphson
a	0.003955805	0.006935996	0.00195503
b	0.001690295	0.002963715	0.000835374
c	0.003627771	0.006360831	0.00179291
d	0.027019899	0.047375925	0.013353721

Geometry	First Order	Second Order
a	0.004336035	0.00230562
b	0.001852765	0.000985179
c	0.003976471	0.002114427
d	0.029617041	0.015748406

TABLE 3.5: Computational time in seconds of the distance computation for different geometries of Figure 3.15. The mesh used for each geometry contains 100 nodes. The computational time is the sum of the distance computation for the 100 nodes of the mesh

It is clear that among all the implemented methods the Newton-Raphson remains the best in terms of computational cost (Table 3.5). The second order method still provides fast results, then comes the Bent method, the first order method and the hybrid Newton-Raphson. Moreover we have conducted other accuracy and robustness tests and they showed that both the Newton-Raphson and the hybrid Newton-Raphson methods are more robust and provide more accurate results than the three other methods. As it is the fastest one, we have implemented only the Newton-Raphson method in 3D. The other reason is that this method is easy to implement in 3D, unlike the first-order, the second-order and the brent methods.

3.1.2.4 Computing the sign of the distance

So far we have presented methods to compute the distance relatively to NURBS based objects, which is the first step to compute their level-sets. The second step consists in signing the distance in order to check whether the point lies inside or outside the object. If the point is outside the object, then the distance will take a negative sign and vice versa. We propose two methods for signing the distance. The first one consists in defining a point O lying inside the object and computing the scalar product $\mathbf{P_pP}.\mathbf{P_pO}$, P being the query point (node of the computational mesh) and P_p the closest point of P on the object boundary (Figure 3.16).

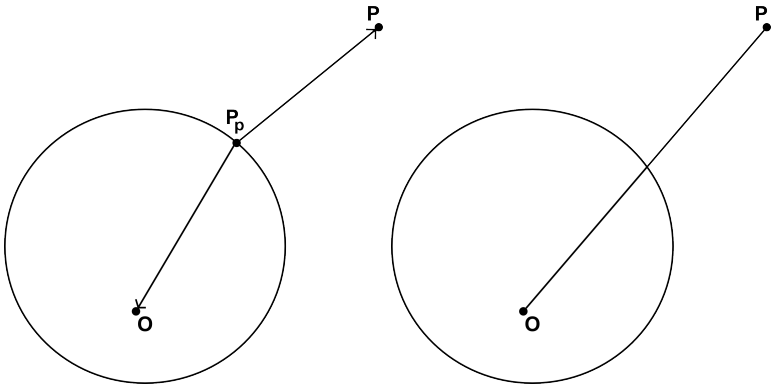


FIGURE 3.16: Scalar product signing method (left) and intesection signing method (right)

If the sign of the obtained scalar product is negative, then it means that the point P is outside the object and the distance takes a negative sign. This method is efficient and easy to implement but its main drawback lies in the fact that it works only for convex objects. The second method is more generic and works for any type of objects. It consists in computing the number of intersections between the edge constituted by the query point P and the inside point O and the object's boundary (Figure 3.16). If the number of intersections is odd, then the distance takes a negative sign.

3.2 Conclusion

In this chapter we have presented a new immersion technique, which is based on NURBS functions. This method bypasses the generation of a surface mesh as the CAD file is directly immersed in the computation. It is clearly complementary with anisotropic mesh adaptation since the adaptation allows to recover the accuracy of the NURBS functions. Moreover it makes the Immersed Volume Method even more generic and makes the build up of a fluid structure application easier.

Computing the level-set to NURBS based objects consists in computing the distance function at each node of the computational mesh. The main steps are to find a good

initial guess for the iterative method and then to perform this iterative method to find the distance. Finaly the distance has to be signed. We have presented several methods to find a good initial guess and have shown that the fastest one is the sampling method. Also the most efficient iterative method is the Newton-Raphson method. We have presented 2D and 3D results of the method, which is fully parallelized. Finally the outline of the chosen algorithm regarding the obtained results takes the following form:

1. points on the NURBS curve (respectively surface) are sampled.
2. then we find the closest of these points to the query point (a node of the computational mesh).
3. compute the distance with the Newton-Raphson method with the closest sampled point as a starting value.
4. sign the distance with the intersection method.

This algorithm has been implemented and used in several CFD applications with complex geometries. This tests are presented in the following chapter and demonstrate the robustness and the efficacity of this method.

Résumé français

Les méthodes d'immersion sont de plus en plus utilisés par la communauté scientifique. Différentes méthodes d'immersion ont été développées comme la méthode d'immersion de frontière [12], la méthode cartésienne [15] ou la méthode d'immersion de volume [14]. Toutes ont pour but de simplifier la mise en place de calculs en intéraction fluide-structure. Dans le logiciel Thost la méthode d'immersion de volume est utilisée. Cette méthode consiste à représenter les objets présents dans les calculs par une fonction level-set, qui est une fonction distance signée.

D'habitude cette fonction distance est calculée par rapport à un maillage surfacique de l'objet lorsque celui-ci possède une forme complexe. Dans ce cas, lorsque le maillage de calcul devient suffisamment fin, la description de l'objet est limitée par la résolution du maillage surfacique initial. Nous proposons une méthode innovante pour immerger les objets dans les domaines de calcul. Plutôt que de calculer leur fonction distance par rapport à leur maillage surfacique, la distance est directement calculée par rapport à leur fichier CAO contenant des fonctions NURBS. Les principales étapes pour calculer la fonction distance par rapport à des objets définis par des NURBS sont premièrement de trouver une valeur initiale judicieuse en vue du calcul de la distance avec un algorithme itératif. Enfin cette distance est signée selon qu'on se trouve à l'intérieur ou à l'extérieur de l'objet.

Dans ce chapitre plusieurs méthodes pour trouver une bonne valeur initiale ainsi que plusieurs méthodes itératives ont été développées et testées. Il en ressort que la méthode sélective la plus rapide est celle de l'echantillonnage et que la méthode itérative la plus robuste est la méthode de Newton-Raphson. Des cas 2D et 3D de la nouvelle méthode d'immersion sont présentées et montrent que la méthode a été entièrement parallèlisée.

Chapter 4

Combining Anisotropic Mesh Adaptation & NURBS Immersed Method

The performance of the new NURBS Immersed Method will be assessed using several 2D and 3D examples. First we show that combining the new immersed method with anisotropic mesh adaptation can lead to a novel, efficient and flexible immersed framework able to handle simple and very complex geometries. Then we combine it with flow solvers based on stabilized finite element method to simulate complex fluid structure interaction problems. The results show that the method is very efficient and robust in particular at high Reynolds numbers using anisotropic meshes with highly stretched elements. Finally we present interesting alternative immersed methods to complete the immersion framework.

4.1 Immersed 2D and 3D simple geometries

First we test the method by immersing simple objects. Indeed, the distance function for the circle and the rectangle can be obtained easily using analytical functions. Therefore, they will be used first to test the implemented algorithm, in particular in the presence of curvatures, sharp angles and singularities. We immerse the CAD descriptions of a circle, a rectangle and a NACA profile in 2D, a sphere and a cube in 3D. We use the computed levelset functions as the mesh criterion.

Figure 4.1 presents the zero isovalues of the immersed objects inside the computational domain. As expected, it reflects the sharp capture of the geometries and the right orientation and deformation of the mesh elements (longest edges parallel to the boundary).

This yields to a great reduction of the number of triangles and consequently to a smaller computational costs. These first results show that the method works properly and that the obtained results are accurate and well respect the geometry of the objects.

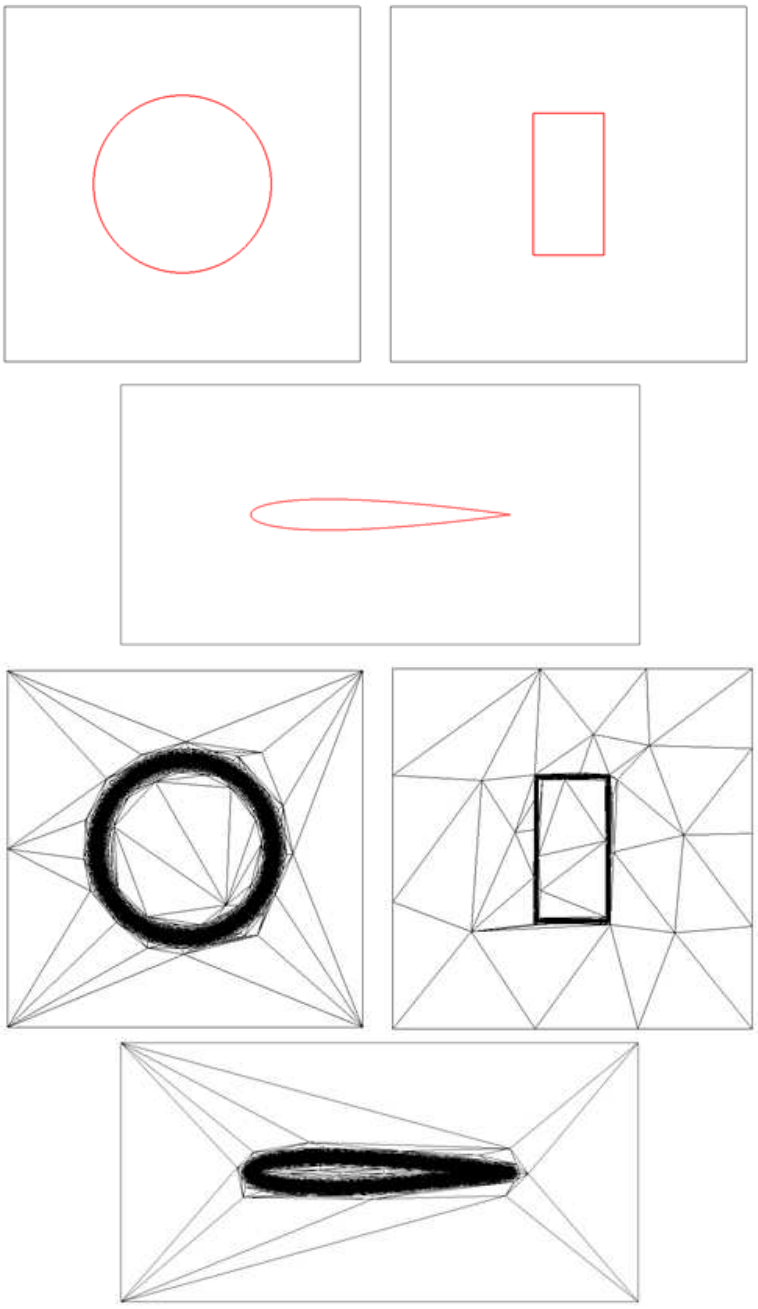


FIGURE 4.1: 2D applications of the Immersed NURBS Method: level-set zero iso-value (top), adapted meshes (bottom)

The extension of the method to deal with 3D objects described this time by NURBS surfaces is tested on a sphere and a cube immersed inside a larger domain. Figure 4.2 shows the zero-isovalues of the computed levelset functions and several cuts in the planes presenting the obtained meshes at the interfaces. Once again the results prove that the implemented method works well and show that combining the new immersed

method with anisotropic mesh adaptation leads to a very practical and accurate tool for immersed methods.

Taking a closer look at the mesh near the interfaces, we can detect the good orientation of the elements with the stretching in the right direction. This demonstrates the ability of the algorithm to work under the constraint of a fixed number of nodes and to effectively control the elements size, orientation and location. Details of the mesh adaptation mesh are given in the next chapter.

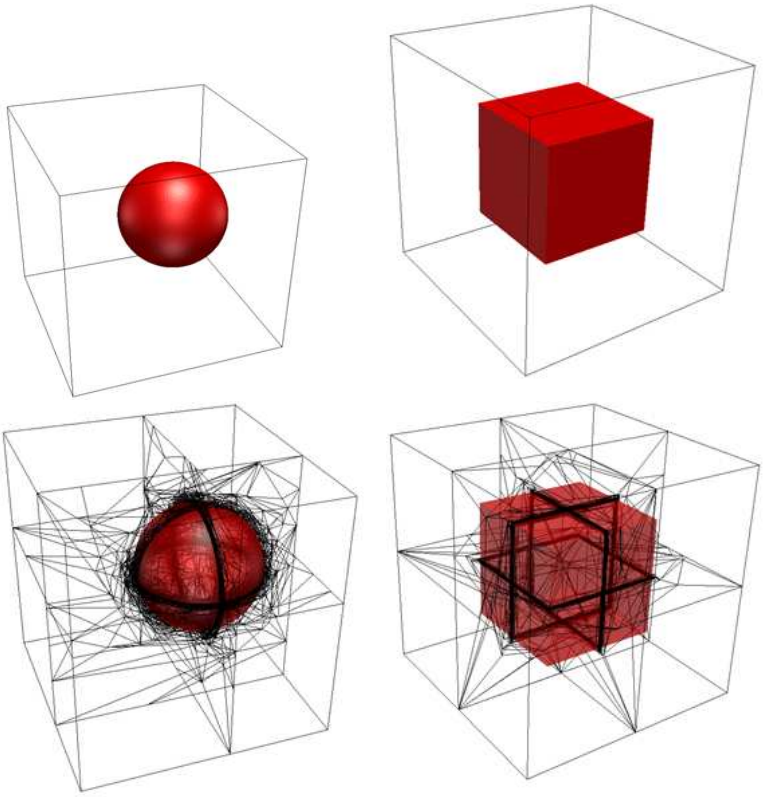


FIGURE 4.2: 3D applications of the Immersed NURBS Method: level-set zero iso-value (top), adapted meshes (bottom)

4.2 Immersed 3D complex geometries

In this section, we test the immersed method on complex geometries: a ship hull and a large airship. Two difficulties must be underlined. The first is clearly the edge of the ship hull while the second is the presence of the hole all along the airship. Note also that both geometries are described this time by several NURBS surfaces.

The same algorithm is applied iteratively on both geometries: (i) distance function computation using NURBS, (ii) sign determination and (iii) anisotropic mesh adaptation. The obtained results are shown in Figure 4.3. As expected, the algorithm progressively detects and refines the mesh at the interfaces leading to a well respected shape in terms

of curvature, angles, etc. All the small details in the given geometries are captured accurately. These observations reflect the ability of the anisotropic mesh adaptation algorithm to automatically adjust the shape and orientation of the elements while optimizing their numbers. For instance, the singularity of these edges could not be recovered without an accurate distance computation and anisotropic refined mesh adaptation.

It is worth mentioning that using both NURBS and anisotropic mesh adaptation is complementary. As mentioned previously, immersed objects are usually surface meshes. Therefore the anisotropic mesh adaptation can be limited by the facetization of the object, i.e. the accuracy of the surface mesh file. By immersing NURBS objects we overcome this issue as the object geometry is kept analytical. Thus the anisotropic mesh adaptation reaches its full potential.

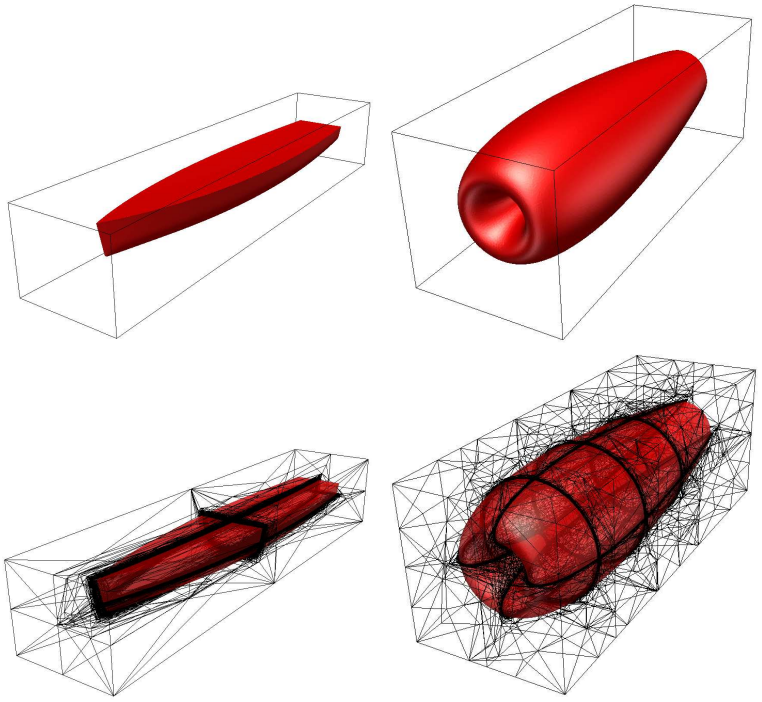


FIGURE 4.3: 3D applications of the Immersed NURBS Method: level-set zero iso-value (top), adapted meshes (bottom)

We present in table 4.1 the computational time taken to compute the distance function of the ship hull. We compare several techniques and we use different number of cores (1, 2, 4 and 8) also to test the implementation in a parallel environment. First, we notice that the algorithm works well in parallel and shows a good scalability. Secondly, we compare the present method to the computation of the distance function obtained by immersing a surface mesh (i.e. STL file). Even though the comparison is not fair since the execution time to obtain the surface mesh is not counted and the quality of the surface mesh remains unclear, the purpose of this comparison still gives us an idea on the potential of the method and the possibilities for improvement. However, to make the comparisons fair, we immersed first the ship hull inside a smaller domain using the NURBS, and then we interpolate the obtained distance function on this refined mesh to the larger

# cores	NURBS	Surface Mesh	NURBS + Interpolation
1	138.10	13.37	2.72
2	70.92	6.99	2.23
4	43.14	3.53	2.12
8	22.30	2.03	0.70

TABLE 4.1: Computational time in seconds of the distance calculation of the ship hull immersed with an IGES file, a STL file and the interpolation method. The computational mesh for this case is made of $102,040$ nodes.

computational domain. In the latter case, the cost of this method referred as NURBS + Interpolation becomes negligible and interesting for practical CFD applications. This interpolation method is more detailed in section 4.4.1.

4.3 CFD applications

In this section we investigate two 3D CFD cases. The aim is to show the capability of the Immersed NURBS Method to deal with complex shapes and to be used in a finite element environment [61]. The first case presents the flow around an airship and the second case the rotation of a propeller in water.

4.3.1 Flow around an airship

The objective of this test case is to demonstrate the utility of the Immersed NURBS Method. Indeed, combined with flow solvers it allows to easily and accurately deal with complex fluid structure interaction problems. Therefore, we consider a turbulent flow past an immersed large scale airship (Figure 4.4). Air is injected in the cavity at $30m/s$, inducing a Reynolds number of 4.10^7. This case is also very interesting considering the dimensions of the airship and the cavity. the cavity is $700m$ long and the airship $75m$. Therefore this case demonstrates the capabilities of the Immersed NURBS Method, the stabilized solvers and the anisotropic mesh adaptation method to deal with large scales. This 3D computations have been obtained using 64 2.4Ghz Opteron cores. The air flow around the airship is quite complex and interesting; i.e. the study of different airfoils and their positions to optimize the aerodynamic design is made possible. A number of turbulent vortices can be observed around and in the wake of the object. All these observations are highlighted by the streamlines in Figure 4.5. Moreover, we can clearly see on the vertical cuts that the solid region satisfies the zero velocity and, hence, the no-slip condition on the extremely refined interface. The airship slows down the air circulation on the surface and influences the main air circulation along the hole.

Note also in Figure 4.6 the concentration of the resolution not only along all the boundary layers but also at the detachment and in the wake regions. This reflects well the anisotropy of the solution caused by the discontinuity of the boundary conditions and the nature of the flow. The elements far from the immersed solid are mostly isotropic and increase in size as the velocity gradient decreases. Again, this reflects and explains why, for a controlled number of nodes, the mesh is naturally and automatically coarsened in that region with the goal of reducing the mesh size around the boundaries and in the wake regions.

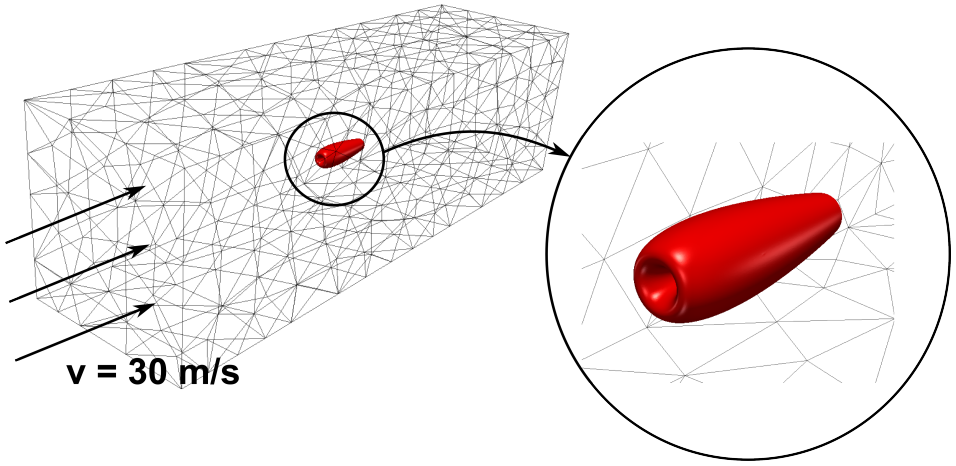


FIGURE 4.4: Configuration of the airship case

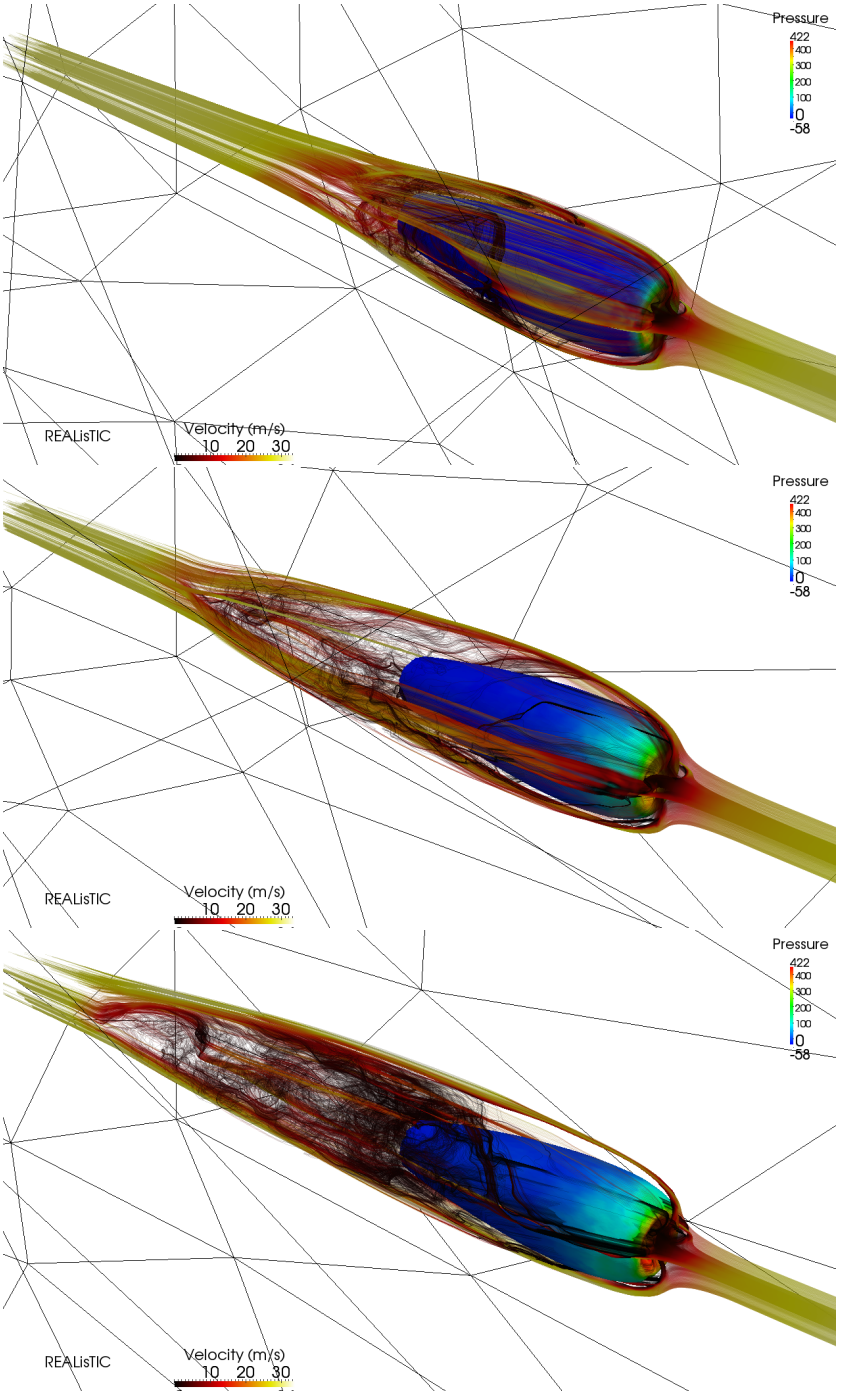


FIGURE 4.5: Snapshots of the streamlines around an airship described by NURBS surfaces at different times

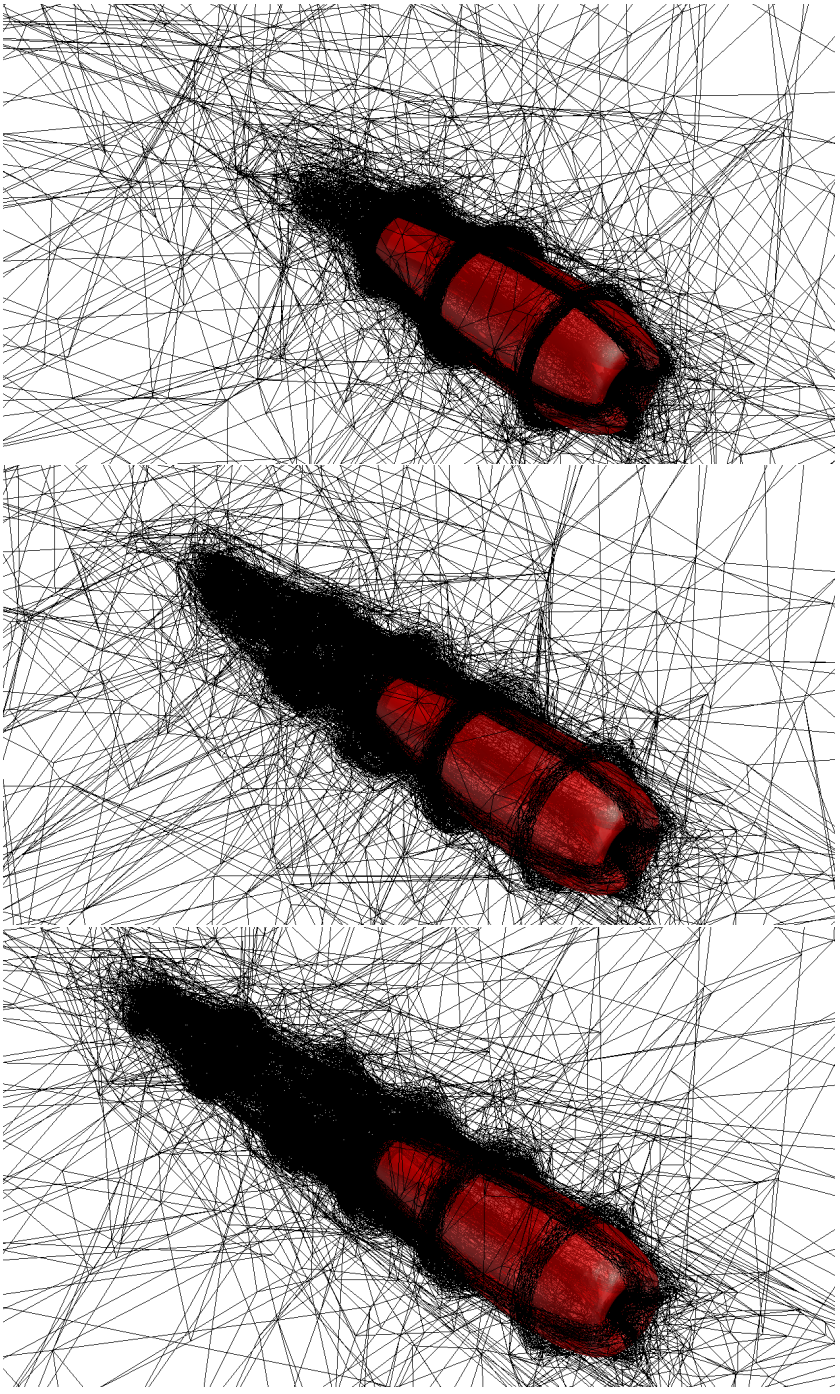


FIGURE 4.6: Snapshots of the adapted mesh around around an airship described by NURBS surfaces at different times

4.3.2 Flow induced by the rotation of a propeller

The second test case is the rotation of a propeller in water (Figure 4.7). The objective here is to deal with a moving object. This case demonstrates the advantage of the Immersed Volume Method to facilitate the build up of a fluid structure application. Therefore the object is represented by its level-set and this one is updated every time step as the propeller rotates. Unlike the body fitted methods, there is no need of fitting the mesh of the domain with the object geometry at each time step. Moreover no surface has been created for the propeller as it has been immersed directly from its CAD file. The propeller has a diameter of $0.55m$ and rotates with an angular velocity of $10rad/s$. Again here the level-set of the propeller has been computed on an extremely dense adapted mesh as an initial step and then the level-set is interpolated to the computational mesh by the interpolation method at each time step. Here the rotational velocity of the solid is imposed by using a Dirichlet condition.

Figure 4.8 shows streamlines around the object at a certain time of the computation. The flow is induced by the rotation of the propeller and has an helicoïdal profile. We can also notice how the mesh is refined and adapted around the interface. As the mesh criteria contains the level-set as well as the velocity, the mesh is also adapted on the velocity.

Therefore these two CFD applications have demonstrated that the NURBS Immersion Method is effective and works well. To do so we have been able to get the analytical geometries defined by NURBS functions from CAD files. Then we have used these NURBS functions to compute the signed distance function of the objects in an optimzed mesh. These two operations have been achived by using and linking a C++ library (GoTools) to Cimlib. Afterwards we have interpolated this distance function to the computational mesh and finally we have adapted the mesh anisotropically on the interface of the object, thus recovering the NURBS accuracy.

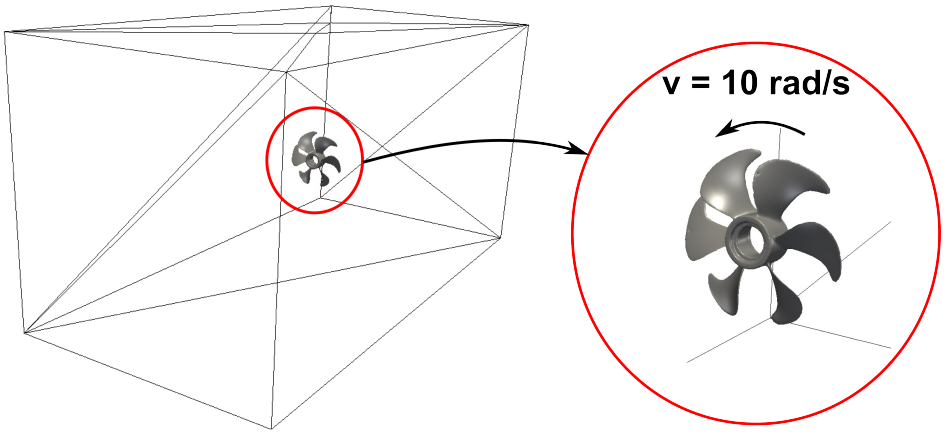


FIGURE 4.7: Configuration of the propeller case

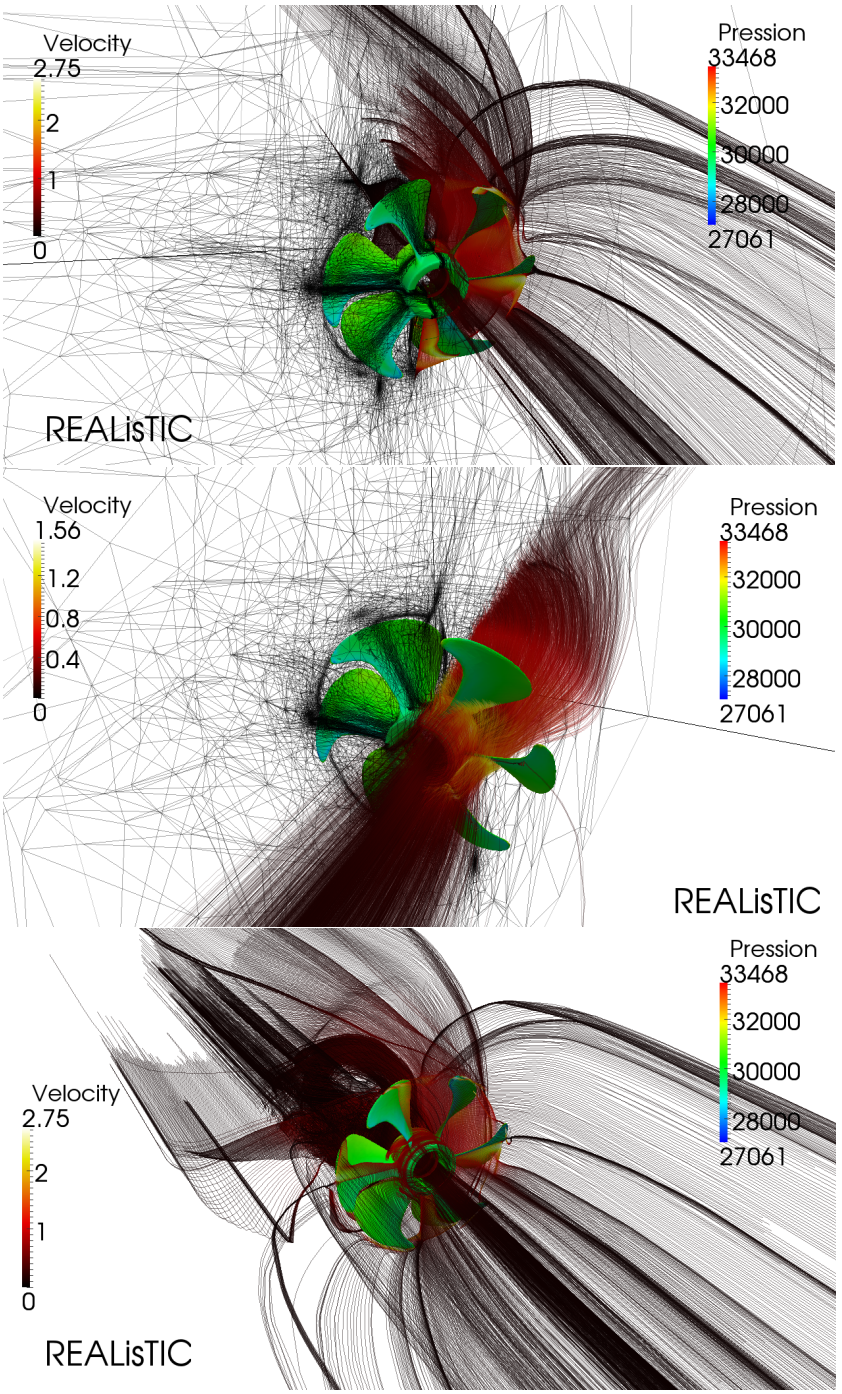


FIGURE 4.8: Snapshots of the streamlines around a propeller described by NURBS surfaces

4.4 Alternative methods

The basic idea of the NURBS Immersion Method is to develop a more generic way for immersing objects and to make the set up of the computation easier, which is one of the main advantage of the Immersed Volume Method. The other advantage of the NURBS Immersion Method is its accuracy. Indeed the level-set of the object is more accurate when computed relatively to NURBS than to a surface mesh. Moreover the accuracy of the level-set is crucial with the Immersed Volume Method as the mixing laws are done at the objects interfaces. Thus immersing NURBS makes the problem smoother. And the anisotropic mesh adaptation allows to recover this sharp interface and thus the accuracy. Therefore computing the level-set relatively to NURBS and adapting the mesh at the interface are complementary.

However the different methods tested to immerse NURBS based objects are slower than computing the distance function to a surface mesh (table 4.1). For this purpose we have developped other ways of immersing objects in the computations.

4.4.1 Interpolation method

The first alternative method to immerse object is to interpolate its level-set from a preadapted mesh to the computational mesh. As the computation of the level-set relatively to NURBS needs further investigations in order to reduce the computational time, the method proposed here is a good alternative. The idea is to immerse a NURBS based object in an optimized mesh. By optimized mesh we mean that the dimensions of the mesh are just large enough to immerse the entire object. Then we adapt the mesh on the level-set until the accuracy is good enough. Finally the level-set will be interpolated from this initial adapted mesh to the computational mesh by using a parallelized interpolation method. Figure 4.9 shows all the steps of this optimized immersed method.

In fact the interpolation method consists in interpolating a field, the level-set in our case, from the initial mesh to the final one. The level-set is P1, i.e. linear on an element and continuous. Therefore we have to find to which element of the initial mesh belongs a node of the final mesh, interpolate the value and repeat the process for every node of the final mesh. Of course this type of algorithm is not optimized as it is of order $O\left(n^2\right)$. A hierarchization of the elements of the initial mesh directly leads to an algorithm of order $O\left(n \log\left(n\right)\right)$. The elements are recursively stored into boxes, therefore we just test recursively if the node of the final mesh belongs to a box, thus reducing the number of tests between the nodes and the elements. This method is parallelized in Cimlib by testing the boxes on the different mesh partitions [62].

To emphasize the interest of the interpolation method we have tested its effectiveness on the ship hull case. The level-set of the ship hull has been first computed on a mesh

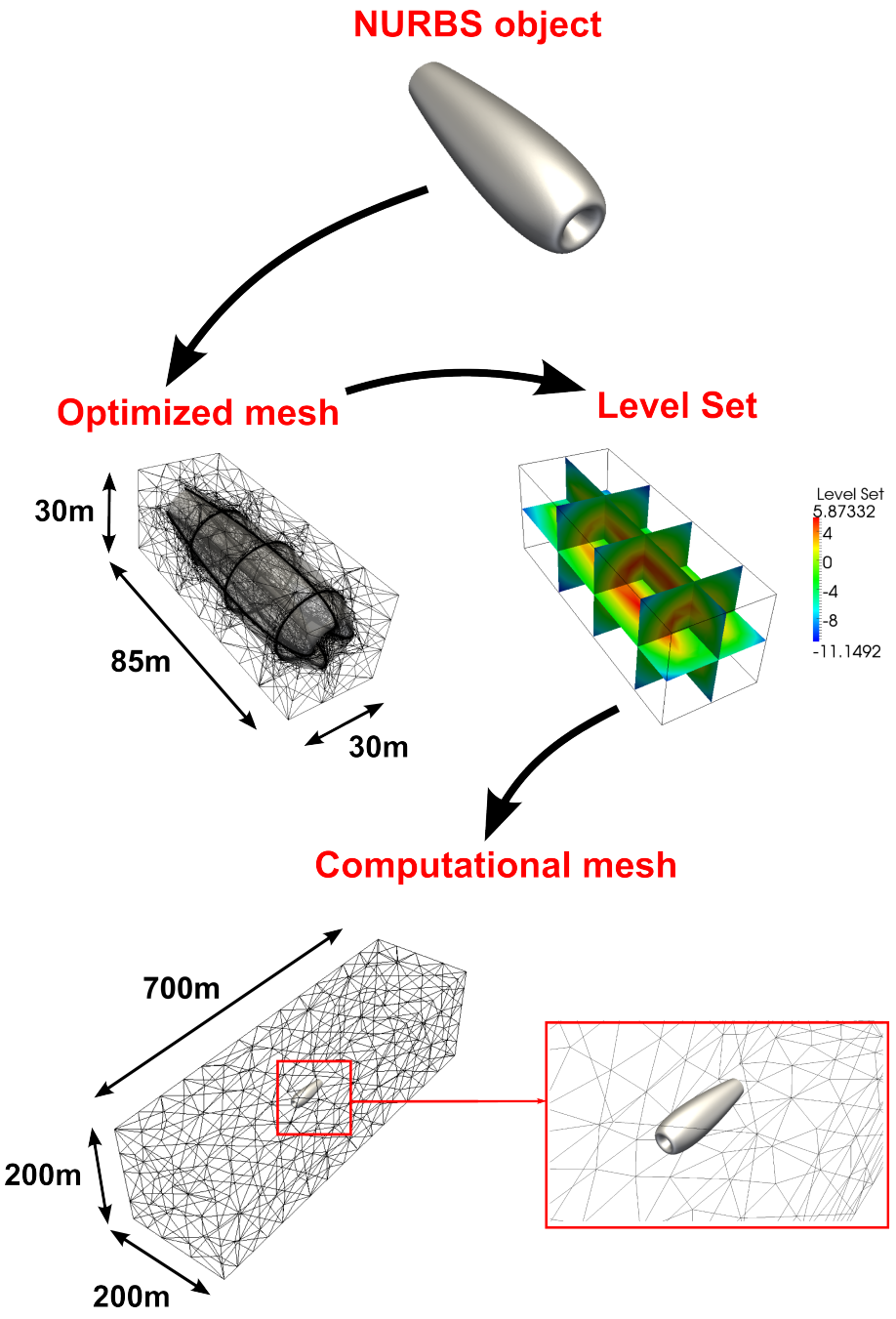


FIGURE 4.9: Interpolation method scheme: the level-set is first computed relatively to the NURBS object in an optimized mesh and then interpolated in the computational mesh

of optimal dimensions. The mesh has been adapted anisotropically until recovering a satisfying interface of the ship hull. The final adapted mesh is made of around $800,000$ elements. Then we have interpolated the level-set on a larger mesh ($600,000$ elements) to compare the computational time of the interpolation method with the one of the classic NURBS Immersed Method presented before. The results are shown in table 4.1, corresponding to the label NURBS + Interpolation. It is clear that interpolating the level-set to the computational mesh is faster (up to 50 times) than recomputing the level-set relatively to the NURBS object. It is worth mentioning that the computational time of the interpolation method decreases with the number of cores used. The meshes that have been used are presented in Figure 4.10.

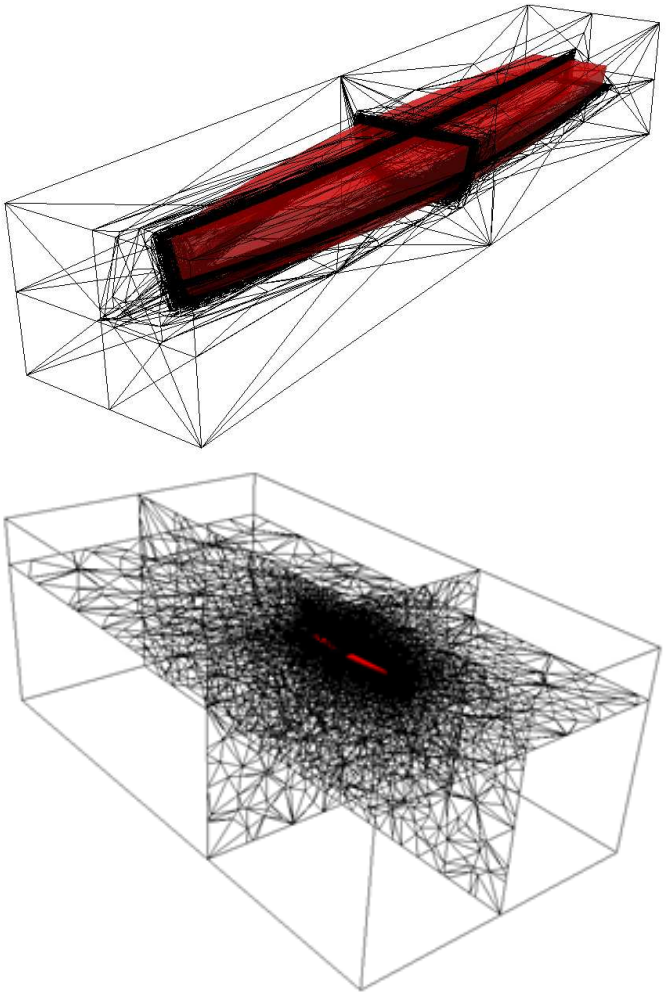


FIGURE 4.10: Meshes used for the level-set interpolation of the ship hull: optimized mesh (top) and computational mesh (bottom)

4.4.2 Point clouds

The second alternative method to immerse objects is to compute the distance function relatively to a point cloud. In fact 3D scanning of objects has expanded since the 80's

and is more and more used by industrials to catch the geometry of complex objects. Given an object, its geometry is recovered by laser scanning, leading to a point cloud. Point clouds are mostly used for graphic visualization and representation in Computer Graphics [63], but also for segmentation, feature extraction and surface reconstruction. The surface can be reconstructed explicitly [39, 64, 65] or implicitly [66]. The reader is invited to read [67] for a more detailed review of the existing methods. 3D scanned point clouds lead to a dense set of points. Only the coordinates of the points are stored. As the amount of data is dense (potentially billions of points), recovering the level-set from a point cloud is a very attractive method with a lot of potential in terms of accuracy.

We state the problem as computing the level-set to a point cloud, which provides the coordinates of the scattered points, and a normal of the surface for each point. Thus it consists in computing the signed distance of every node of the computational mesh to the point cloud. A simple idea to do this is to find the closest point X_i of the point cloud Ω for each node P of the mesh, and compute the distance by the following formula:

$$d(P,\Omega) = \min_i d(P,X_i) \,.\mathbf{n_i}.\frac{\mathbf{PX_i}}{\|\mathbf{PX_i}\|} \qquad ,\mathbf{X_i} \in \mathbf{\Omega} \tag{4.1}$$

with $\mathbf{n_i}$ the normal at point X_i. Unfortunately this formula leads to stiff level-sets and encounters issues when the geometry has singularities (Figure 4.11). In fact if the point cloud is not dense enough around the singularities it can occur that the closest point found in the point cloud has not the good normal, leading to a wrong level-set computation. Therefore we have decided to use a formula giving a smoother results, and less dependent of the quality of the point cloud:

$$d(P,\Omega) = \frac{w_1 s_1 + w_2 s_2}{w_1 + w_2} \tag{4.2}$$

$$\begin{aligned} w_i &= d(P,X_i)^{-p}, \quad && i \in [1,2] \\ s_i &= \mathbf{PX_i}.\mathbf{n_i}, && i \in [1,2] \end{aligned}$$

For every node of the computational mesh, the distance is computed by considering the two closest points of the point cloud. A weight is attributed to each point relatively to the inverse of the distance. We point out that p is a user parameter, Figure 4.12 shows the influence of this parameter. The level-set has been computed for different values of p relatively to the point cloud of a square. The point cloud is composed of 4 points. We can see that p has an influence on the results. The singularities are better fitted when a higher value of p is used.

Despite the fact that the two-points formula gives smoother level-sets and is less dependent on the quality of the point cloud than the one-point formula (Figure 4.11), we can not ensure a good quality of the level-set in unfavorable cases. Therefore we enlarge the

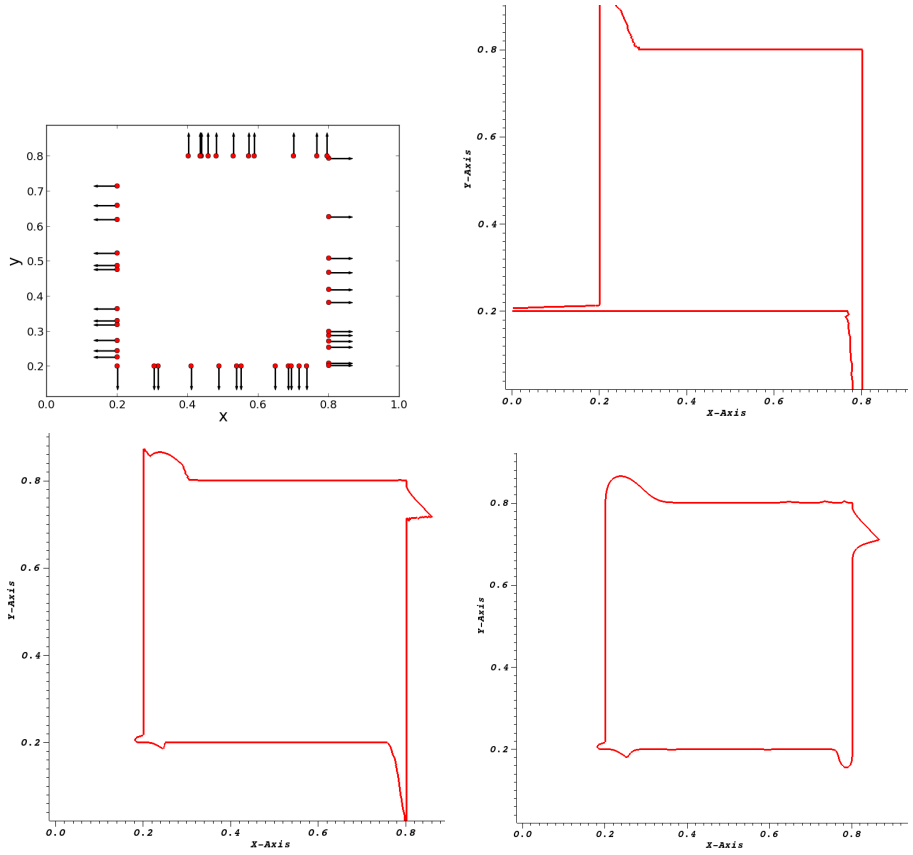


FIGURE 4.11: Point cloud of a square made of 50 points (top left) and the zero isovalue of the level-set with different methods: 1-point method (top right), 2-points method (bottom left), n-points method (bottom right)

method by taking under consideration for each node of the computational mesh the n closest points of the point cloud. The n closest points are the ones present in the circle (in 2D) or the sphere (in 3D) centered on the closest point. Thus the method consists in first finding the closest point of the point cloud to the node P, and then considering the n points inside the circle or the sphere in the formula. Only the points having a normal differing from a certain value are kept. The radius of the circle (sphere) is computed relatively to a caracteristic length of the object provided by the user. The outline of the algorithm is detailed in Algorithm 8.

$$d\left(P,\Omega\right)=\frac{\sum_{i=1}^{n}d_i^{-p}\mathbf{PX_i}.\mathbf{n_i}}{\sum_{i=1}^{n}d_i^{-p}} \tag{4.3}$$

Figure 4.11 shows that the formula is better suited to compute the level-set of a point cloud object when there are singularities. Comparing to the other formulas, this one leads to a closed object, which is essential for numerical applications. Obviously the level-set obtained in Figure 4.11 does not recover perfectly the initial geometry, but it is worth mentioning that the better the quality of the point cloud, the closer to the initial geometry the level-set. That means that here the quality of the point cloud does not let

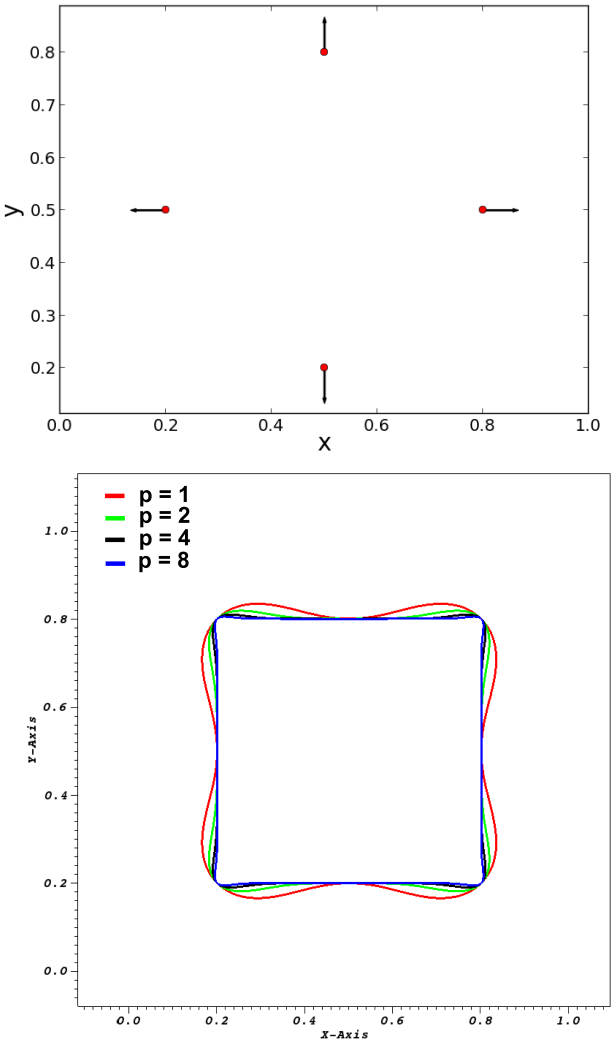


FIGURE 4.12: Point cloud of a square made of 4 points (top) and the zero isovalue of the level-set for different values of p (bottom)

Algorithm 8 n-points method for point cloud distance computation

Find the closest point of the point cloud from P
Compute the radius of the circle (2D) $r = \frac{2}{\frac{n}{2\Pi L}}$, or the sphere (3D) $r = \frac{2}{\sqrt{\frac{n}{4\Pi L^2}}}$
Find the n points $\mathbf{X_i}$ contained in the circle or the sphere
for each point $\mathbf{X_i}$ **do**
  for each point $\mathbf{X_j}$ **do**
    if $\mathbf{n_i}.\mathbf{n_j} < \epsilon, \mathbf{i}, \mathbf{j} \in [\mathbf{1}, \mathbf{n}], \mathbf{i} \neq \mathbf{j}$ **then**
      Remove $\mathbf{X_j}$
    end if
  end for
end for
Compute the distance with equation (4.3)

a good recover of the geometry. Then the proposed method has been used with point clouds of better quality, leading to the results presented in Figures 4.13 and 4.14. We

can see that the method captures extremelly well all the curvatures of the objects (e.g. the circle and the sphere), but still does not recover perfectly the singularities (e.g. the square and the cube). The anisotropic mesh adaptation has been used in order to get a good description of the computed level-set functions.

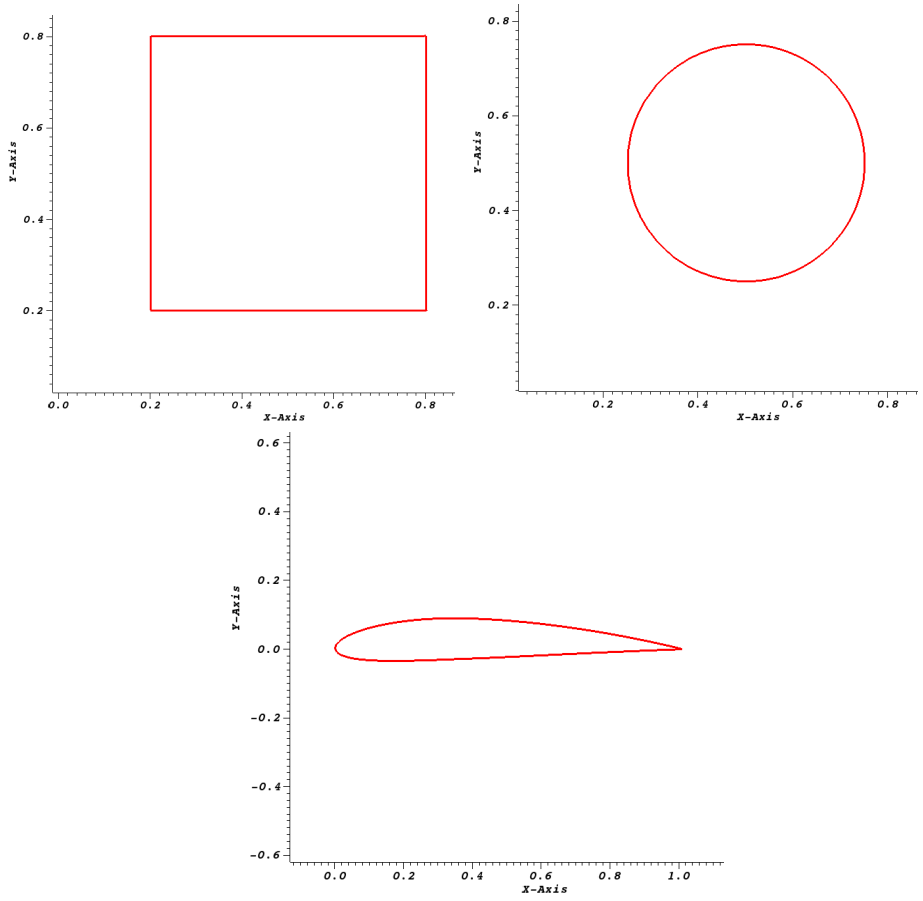


FIGURE 4.13: Zero isovalue of the levelsets of a square (top left), a circle (top right) and NACA profile (bottom). The point clouds of these three objects have $1,000$, 250 and $10,000$ points respectively

4.5 Conclusion

In this chapter, the performance of the new NURBS Immersed Method has been demonstrated. 2D and 3D examples are provided showing that the method is capable of handling simple geometries such as spheres or cubes. Then a comparison is made with the standard level-set computation of the IVM (relatively to a surface mesh) on the ship hull case. The results show that the new method still needs to be improved. Therefore we propose an alternative method to accumulate the advantage of the NURBS Immersed Method and keep a reasonable computational time. The level-set of the NURBS based object is first computed in an optimized mesh and then interpolated in the computational mesh with a hierarchical interpolation method.

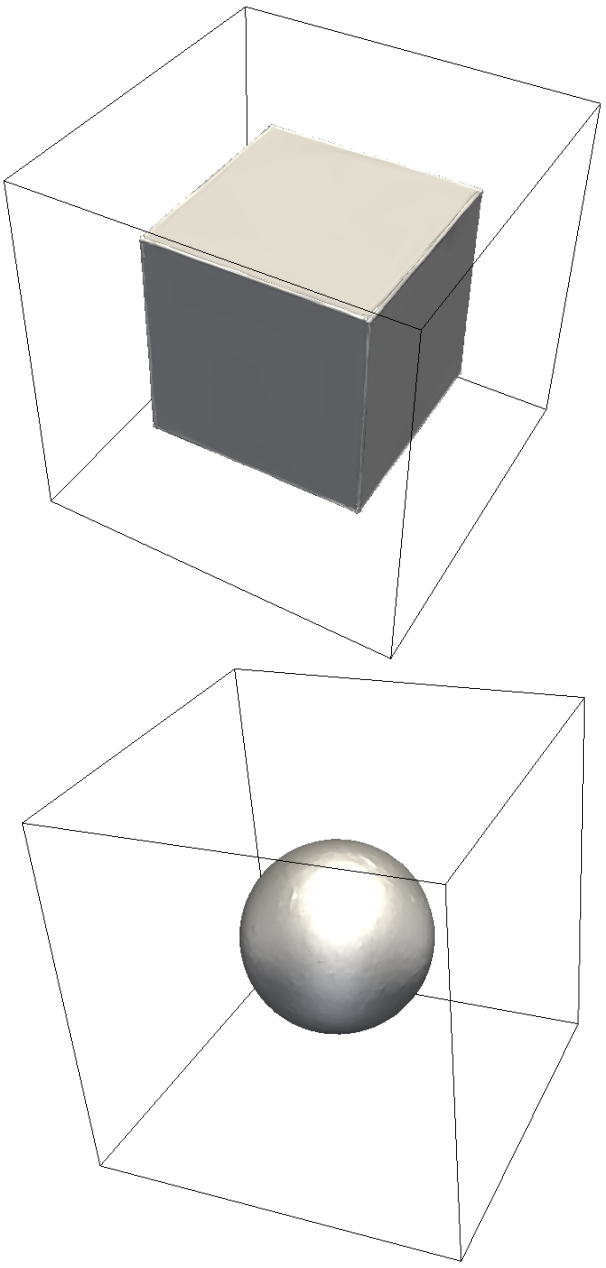


FIGURE 4.14: Zero isovalue of the levelsets of a cube (left) and a sphere (right). The point clouds of these two objects have $250,000$ and $10,000$ points respectively

Two CFD applications have been investigated: the flow around a ship hull and the rotation of a propeller in water. The first case shows the capibility of the Immersed NURBS Method, the stabilized solvers and the anisotropic mesh adaptation to deal with highly turbulent flows into large scales. The second one validates the flexibility of the Immersed NURBS method to build up a CFD application and handle a moving object. Another alternative method is provided in order to immerse objects. As point clouds are gaining popularity in the scientific and engineering community to represent complex geometries, we propose a new method to compute level-set functions directly to point clouds. The capability of the method to represent simple shapes has been shown

and further developments are needed in order to improve the method for complex shapes, especially shapes having singularities.

Résumé français

Dans ce chapitre la nouvelle méthode d'Immersion de NURBS est testée et mise à l'épreuve. Dans un premier temps des cas de géométries simples 2D et 3D sont présentés, démontrant la capacité de la méthode à immerger des objects simples. Ensuite des formes plus complexes sont utilisées pour évaluer les performances de la nouvelle méthode en terme de temps de calcul. Ainsi la méthode est comparée à l'immersion classique (calcul de distance par rapport à un maillage surfacique). Les résultats montrent que la méthode d'Immersion de NURBS est plus lente que la méthode d'immersion classique. La méthode nécessite donc des études et des développements supplémentaires afin d'être améliorée. Nous proposons une méthode alternative permettant de cumuler la précision et la flexibilité de la méthode d'Immersion de NURBS et un temps de calcul rapide. Cette méthode consiste à immerger dans un premier temps l'objet à base de NURBS dans un maillage optimisé (dimensions du domaine adaptées à la géométrie de l'objet et maillage adapté à l'interface). Ensuite la level-set calculée dans le maillage optimisé par la méthode d'Immersion de NURBS est transportée au maillage de calcul par une méthode de transport hiérarchique.

Afin de valider cette approche, deux cas d'interaction fluide-structure sont présentés. Le premier concerne l'écoulement turbulent d'air autour d'un ballon dirigeable. L'intérêt de ce premier cas est de démontrer la capacité de la nouvelle méthode ainsi que des solveurs stabilisés et de la méthode d'adaptation de maillage anisotrope à être opérationnels avec des écoulements très turbulents dans des grands domaines (plusieurs centaines de mètres). Le second cas montre la rotation d'une hélice dans de l'eau. Cette fois-ci la difficulté du cas réside dans le fait que l'objet bouge. Ce cas valide donc la flexibité de la nouvelle méthode pour mettre en place des problèmes d'interaction fluide-structure. La level-set est simplement réactualisée au fur et à mesure que l'objet se déplace.

Enfin une autre méthode alternative est présentée. L'interêt de la communauté scientifique et des ingénieurs à propos des nuages de points ne cessant de grandir et les technologies évoluant, nous proposons une nouvelle méthode pour immerger directement les nuages de points dans les calculs. Le potentiel de cette méthode réside dans la densité des données (potentiellement plusieurs milliards de points) pouvant mener à une excellente précision. La méthode a été testée sur des cas 2D et 3D simple et il a été montré qu'elle nécessite des améliorations, en particulier lorsque les objets présentent des singularités.

Part II

Part B

Chapter 5

Stabilized finite element methods for solving coupled problems

We remind that the target of this thesis is to simulate turbulent problems coupled to heat transfers. Therefore in this chapter we present the governing equations used to model numerically turbulent flow problems coupled to heat transfer into large domains. We also introduce the stabilized finite element methods which are crucial in the case of convection dominated problems (high Reynolds and Peclet numbers) and anisotropic mesh adaptation. The stabilized solvers presented thereafter improve the stabilizaty of the solution, remove spurious oscillations and control the numerical shocks through the addition of extra residual terms to the standard Galerkin formulation. Two validation cases are shown in order to demonstrate the capabilities of the solvers to simulate complex problems and handle highly streched elements produced by the anisotropic mesh adaptation. The first case is the heating of four ingots by hot air in a 2D furnace. The second case shows a real application, which is the heating of six metal ingots in a 3D industrial furnace. Comparisons are made with and without anisotropic mesh adaptation in order to highlight the interest and the performance of the method.

5.1 Governing equations

Let $\Omega \subset \mathbb{R}^d$, $d = 2,\ 3$, be the spatial computational domain with boundary $\partial\Omega$. In order to compute the motion of an unsteady, incompressible, non-isothermal flow with

buoyancy forces, one has to solve the coupled non-linear system provided by the Navier-Stokes equations including the Boussinesq approximation:

$$\boldsymbol{\nabla} \cdot \mathbf{u} = 0 \qquad \text{in } \Omega \tag{5.1}$$

$$\rho(\partial_t \mathbf{u} + \mathbf{u} \cdot \boldsymbol{\nabla}\mathbf{u}) - \boldsymbol{\nabla} \cdot (2\mu\, \boldsymbol{\varepsilon}(\mathbf{u}) - p\mathbf{I_d}) = \rho_0 \beta (T - T_0)\, \mathbf{g} \qquad \text{in } \Omega \tag{5.2}$$

$$\rho C_p(\partial_t T + \mathbf{u} \cdot \boldsymbol{\nabla} T) - \boldsymbol{\nabla} \cdot (\lambda \boldsymbol{\nabla} T) = f - \boldsymbol{\nabla} \cdot \mathbf{q_r} \qquad \text{in } \Omega \tag{5.3}$$

where $\mathbf{u}$ is the velocity vector, p the pressure and T the temperature. Equation (5.1) is the expression of the incompressibility constraint. Equation (5.2) that describes the momentum conservation features the density ρ, the dynamic viscosity μ, the deformation-rate tensor $\boldsymbol{\varepsilon}(\mathbf{u}) = (\boldsymbol{\nabla}\mathbf{u} + {}^t\boldsymbol{\nabla}\mathbf{u})/2$, the reference density and temperature ρ_0 and T_0, the thermal expansion coefficient β and the gravitational acceleration $\mathbf{g}$. Eventually, equation (5.3) denotes the energy conservation and it involves the constant pressure heat capacity C_p, the specific thermal conductivity λ, a volume source term f and the heat radiative flux $\mathbf{q_r}$.

The turbulent aspect of flows in furnaces may require, to reduce the computational cost, the use of dedicated models to compute the flow field. In the present work, we solve the Reynolds-averaged Navier-Stokes problem derived from the equations (5.1)-(5.3) and we resort to the standard $k - \varepsilon$ model to close the system [68, 69]. The RANS equations read:

$$\boldsymbol{\nabla} \cdot \mathbf{u} = 0 \qquad \text{in } \Omega \tag{5.4}$$

$$\rho(\partial_t \mathbf{u} + \mathbf{u} \cdot \boldsymbol{\nabla}\mathbf{u}) - \boldsymbol{\nabla} \cdot (2\mu_e\, \boldsymbol{\varepsilon}(\mathbf{u}) - p_e\, \mathbf{I_d}) = \rho_0 \beta (T - T_0)\, \mathbf{g} \qquad \text{in } \Omega \tag{5.5}$$

$$\rho C_p(\partial_t T + \mathbf{u} \cdot \boldsymbol{\nabla} T) - \boldsymbol{\nabla} \cdot (\lambda_e \boldsymbol{\nabla} T) = f - \boldsymbol{\nabla} \cdot \mathbf{q_r} \qquad \text{in } \Omega \tag{5.6}$$

For sake of simplicity, we kept the same notation for the averaged values of the unknowns such as the velocity $\mathbf{u}$, the effective pressure p_e and the temperature T. The system (5.4)-(5.6) features the effective viscosity μ_e and the effective thermal conduction λ_e which are given by:

$$\mu_e = \mu + \mu_t \quad \text{and} \quad \lambda_e = \lambda + \frac{C_p \mu_t}{\mathrm{Pr}_t} \tag{5.7}$$

with $\mathrm{Pr}_t = 0.85$ the turbulent Prandtl number. The turbulent viscosity μ_t in expression (5.7) is a function of the turbulent kinetic energy k and the turbulent dissipation ε that reads:

$$\mu_t = \rho C_\mu \frac{k^2}{\varepsilon} \tag{5.8}$$

with C_μ an empirical constant usually equal to 0.09. To assess μ_t, the introduced variables k and ε are computed using two transport equations that read:

$$\rho\left(\partial_t k + \mathbf{u}\cdot\boldsymbol{\nabla} k\right) - \boldsymbol{\nabla}\cdot\left(\left(\mu + \frac{\mu_t}{\mathrm{Pr}_k}\right)\boldsymbol{\nabla} k\right) = P_k + P_b - \rho\varepsilon \qquad \text{in } \Omega \tag{5.9}$$

$$\rho\left(\partial_t \varepsilon + \mathbf{u}\cdot\boldsymbol{\nabla} \varepsilon\right) - \boldsymbol{\nabla}\cdot\left(\left(\mu + \frac{\mu_t}{\mathrm{Pr}_\varepsilon}\right)\boldsymbol{\nabla} \varepsilon\right) = \frac{\varepsilon}{k}(C_{1\varepsilon}P_k + C_{3\epsilon}P_b - C_{2\epsilon}\rho\varepsilon) \qquad \text{in } \Omega \tag{5.10}$$

In equations (5.9) and (5.10), P_k represents the production of turbulent kinetic energy due to the mean velocity gradients, P_b is the production due to the buoyancy effects, Pr_k and Pr_ε are the turbulent Prandtl number for k and ϵ respectively, while $C_{1\varepsilon}$, $C_{2\varepsilon}$ and $C_{3\varepsilon}$ are model constants. The production terms P_k and P_b are modelled as follows:

$$P_k = 2\mu_t(\boldsymbol{\varepsilon}(\mathbf{u}) : \boldsymbol{\varepsilon}(\mathbf{u})) \quad \text{and} \quad P_b = -\frac{\mu_t}{\rho \mathrm{Pr}_g}\mathbf{g}\boldsymbol{\nabla}\rho \tag{5.11}$$

Finally, it remains to assess the real pressure from the effective pressure and the turbulent kinetic energy, which is carried out in the following manner:

$$p = p_e - \frac{2}{3}\rho k \tag{5.12}$$

5.1.1 Radiative transfer model

Thermal radiation plays a key role in many industrial processes, like the heating in industrial furnaces, quenching, etc... Its intensity highly depends on the temperature. Therefore it is crucial to use a proper model to take into account this part of the physics.

5.1.1.1 Gray gas assumption

The gray gas model may often be sufficient for furnace applications since, most of the time, surfaces are fairly rough and, as a result, reflect in a relatively diffusive fashion. Furthermore, if the radiative properties do not vary much across the spectrum then the gray gas simplifications may be valid. According to Modest [70], in the case of a gray medium, the divergence of the heat radiative flux that appears in equation (5.3) or (5.6) relies on the local temperature and the incident radiation as follows:

$$-\boldsymbol{\nabla}\cdot\mathbf{q_r} = \kappa\left(G - 4\kappa\sigma T^4\right) \tag{5.13}$$

where G denotes the incident radiation, κ is the mean absorption coefficient and σ the Stefan-Boltzmann constant.

5.1.1.2 The P-1 approximation

Equation (5.13) clearly establishes the necessity of getting an expression of G in order to assess the divergence of $\mathbf{q}_r$. This can be achieved by considering the radiative transfer equation (RTE) that may be found in [71]. In this work, one resorts to the so-called P-1 radiation model that is the simplest case of the P-N model to express radiation intensity by means of series of spherical harmonics (*cf.* [70, 71] for more details). The use of this approach enables the simplification of the RTE into an elliptical partial differential equation in terms of the incident radiation G as follows:

$$\begin{cases} \boldsymbol{\nabla} \cdot \left(\dfrac{1}{3\kappa} \, \boldsymbol{\nabla} G \right) - \kappa G = 4\kappa\sigma T^4 \quad \text{in} \quad \Omega \\ \dfrac{\partial G_{\mathrm{w}}}{\partial n} = \dfrac{3\kappa\epsilon_{\mathrm{w}}}{2(2-\epsilon_{\mathrm{w}})} \left(4\sigma T_{\mathrm{w}}^4 - G_{\mathrm{w}}\right) \quad \text{in} \quad \partial\Omega \end{cases} \tag{5.14}$$

where subscript w denotes wall quantities, n is the normal to the wall and ϵ_{w} the emissivity of the wall.

5.1.1.3 Radiative properties

In the context of gray-medium assumption, the mean absorption coefficient κ can be derived from the emissivity ϵ of the material using the Bouguer's law which reads:

$$\kappa = -\frac{1}{L_m} \ln(1-\epsilon) \tag{5.15}$$

where L_m is the mean beam length defined as:

$$L_m = 3.6 \frac{\Delta V}{\Delta S} \tag{5.16}$$

For unstructured grids, $\Delta V = \Delta x \Delta y \Delta z$ and $\Delta S = 2(\Delta x \Delta y + \Delta y \Delta z + \Delta z \Delta y)$ are appropriate measures of volume and surface for each simplex of the mesh [71].

In this work we thus use the $P-1$ radiation model. However this model offers a good compromise between accuracy and ease of implementation, it has certain limits. We will not discuss these ones as it is beyond the scope of this work. This study is under investigation in [72].

5.1.2 Boundary conditions

At the inflow boundary, for a prescribed velocity $\mathbf{u}$, the value of k can be computed using:

$$k_{\mathrm{inlet}} = c_{\mathrm{bc}} \cdot |\mathbf{u}|^2 \tag{5.17}$$

where c_{bc} is fixed to 0.02 as an empirical constant. Once k is computed, the value of ε can be prescribed using:

$$\varepsilon_{\text{inlet}} = \frac{C_\mu \cdot k^{3/2}}{L} \tag{5.18}$$

with L, a fixed constant, known as the characteristic length of the model [69]. These computed values of k and ε are extended into the interior domain as initial conditions.

At the outflow, the following homogeneous Neumann boundary conditions are applied:

$$\mathbf{n} \cdot \boldsymbol{\nabla} k = 0 \quad \text{and} \quad \mathbf{n} \cdot \boldsymbol{\nabla} \varepsilon = 0 \tag{5.19}$$

On the rest of the computational boundary a combination of Neumann and Dirichlet conditions is imposed by using the classical wall function introduced in [68] which describes the asymptotic behavior of the different variables near the wall. If the boundary mesh nodes are located in the logarithmic region, we impose the wall shear stress given by :

$$\tau_{\text{w}} = \rho {U^*}^2 \tag{5.20}$$

where U^* is the friction velocity evaluated by solving the equation:

$$\frac{U}{U^*} = \frac{1}{\text{k}} \ln\left(\frac{\rho E \delta}{\mu} U^*\right) \tag{5.21}$$

where U is the tangential velocity, δ is the distance to the wall, k is the Von Karman constant (typically equal to 0.41) and E is a roughness parameter taken equal to 9.0 for smooth walls. Imposing the wall shear stress corresponds to a non-homogeneous Neumann boundary condition for the momentum equation in the tangential direction. The normal component of the velocity is set to zero. The turbulent kinetic energy and its dissipation on the boundary of the mesh are given as functions of the friction velocity [68]:

$$k_{\text{w}} = \frac{{U^*}^2}{\sqrt{C_\mu}} \quad \text{and} \quad \varepsilon_{\text{w}} = \frac{{U^*}^3}{k_{\text{w}} \delta} \tag{5.22}$$

Boundary conditions at a wall for the energy equation are enforced through a temperature wall function similar to that used for the momentum equations. The effective heat flux in the wall function is computed as :

$$q_{\text{w}} = \mathbf{n} \cdot \mathbf{q}_{\text{w}} = \frac{\rho C_p C_\mu^{1/4} k_{\text{w}} (T_{\text{w}} - T)}{T^+} \tag{5.23}$$

where T_{w} is the wall temperature and T^+ is the normalized temperature given in [73].

5.2 VMS: incompressible Navier-Stokes solver

In this section the general time-dependent incompressible Navier-Stokes equations are solved. The stabilizing schemes from a variational multiscale point of view are described and presented [74]. Both the velocity and the pressure spaces are enriched which cures the spurious oscillations in the convection-dominated regime and deals with the pressure instability. The stabilization parameters will be determined rigorously taking into account the anisotropy of the mesh using a directional element diameter.

It is well known that the classical finite element approximation for the flow problem may fail because of two reasons: first the compatibility condition known as the inf-sup condition or "Brezzi-Babuska" condition which requires an appropriate pair of the function spaces for the velocity and the pressure [75–79]; and second the dominance of the convection term [80].

Therefore, a recently developed stabilized finite element method which draws upon features of both mixed [81–83] and stabilized finite element methods [84, 85] is used to solve the incompressible Navier Stokes equations for high Reynolds flows. The proposed method start with a stable mixed formulation made of continuous piecewise linear functions enriched with a bubble function for the velocity and piecewise linear functions for the pressure. This choice of elements is stable at low Reynolds number, when the Stokes flow is dominant. However, for simulating high Reynolds flows, an extension of this method based on the variational multi-scale approach is then applied. A decomposition for both the velocity and the pressure fields into coarse scales and fine scales is used, as depicted in [74]. This choice of decomposition is shown to be favorable for simulating flows at high Reynolds number.

Following [86], we consider an overlapping sum decomposition of the velocity and the pressure fields into resolvable coarse-scale and unresolved fine-scale $\mathbf{u} = \mathbf{u}_h + \mathbf{u}'$ and $p = p_h + p'$. Likewise, we regard the same decomposition for the weighting functions $\mathbf{w} = \mathbf{w}_h + \mathbf{w}'$ and $q = q_h + q'$. The unresolved fine-scales are usually modelled using residual based terms that are derived consistently. The static condensation consists in substituting the fine-scale solution into the large-scale problem providing additional terms, tuned by a local stabilizing parameter, that enhance the stability and accuracy of the standard Galerkin formulation.

Let us consider the functional Sobolev space of functions having square integrable first order derivatives $H^1_s(\Omega)$ in which we are searching the solution in accordance with its regularity:

$$H^1_s = \left\{ w \in H^1(\Omega) | w = s \forall x \in \partial\Omega \right\}$$

$$H^1(\Omega) = \left\{ w \in L^2(\Omega), \|\nabla w\| \in L^2(\Omega) \right\}$$

L^2 is the Hilbert vector space given by:

$$L^2(\Omega) = \left\{ w(x) | \int_\Omega |w|^2 dx < \infty \right\}$$

The enrichment of the functional spaces is performed as follows: $V = V_h \oplus V'$, $V_0 = V_{h,0} \oplus V_0'$ and $Q = Q_h \oplus Q'$, with V, V_h, V', Q, Q_h and $Q' \subset H^1(\Omega)$, and V_0, $V_{h,0}$ and $V_0' \subset H_0^1(\Omega)$. Thus, the mixed-finite element approximation of the time-dependent Navier-Stokes problem can read:

Find a pair $(\mathbf{u}, \mathbf{p}) \in V \times Q$ such that: $\forall\ (\mathbf{w}, q) \in V_0 \times Q$

$$\begin{cases} \left(\rho \partial_t (\mathbf{u}_h + \mathbf{u}'), (\mathbf{w}_h + \mathbf{w}')\right)_\Omega + \left(\rho(\mathbf{u}_h + \mathbf{u}') \cdot \boldsymbol{\nabla}(\mathbf{u}_h + \mathbf{u}'), (\mathbf{w}_h + \mathbf{w}')\right)_\Omega \\ \qquad + \left(2\mu \boldsymbol{\varepsilon}(\mathbf{u}_h + \mathbf{u}') : \boldsymbol{\varepsilon}(\mathbf{w}_h + \mathbf{w}')\right)_\Omega \\ \qquad - \left((p_h + p'), \boldsymbol{\nabla} \cdot (\mathbf{w}_h + \mathbf{w}')\right)_\Omega = \left(\mathbf{f}, (\mathbf{w}_h + \mathbf{w}')\right)_\Omega \\ \left(\boldsymbol{\nabla} \cdot (\mathbf{u}_h + \mathbf{u}'), (q_h + q')\right)_\Omega = 0 \end{cases} \tag{5.24}$$

Here f is a source term which can be the gravity or the Boussinesq term for example. when compared with the standard Galerkin method, the proposed stable formulation involves additional integrals that are evaluated element wise. These additional terms, obtained by replacing the approximated $\mathbf{u}'$ and p' into the large-scale equation, represent the effects of the sub-grid scales and they are introduced in a consistent way to the Galerkin formulation. All of these terms are multiplied by stabilizing parameters and enable to overcome the instability of the classical formulation arising in convection dominated flows and to deal with the pressure instabilities.

To derive the stabilized formulation, we first solve the fine scale problem, defined on the sum of element interiors and written in terms of the time-dependent large-scale variables. Then we substitute the fine-scale solution back into the coarse problem, thereby eliminating the explicit appearance of the fine-scale while still modelling their effects. At this stage, two important remarks have to be made in order to deal with the time-dependency and the non-linearity of the momentum equation of the subscale system:

i) the convective velocity of the non-linear term may be approximated using only large-scale part so that $(\mathbf{u}_h + \mathbf{u}') \cdot \boldsymbol{\nabla}(\mathbf{u}_h + \mathbf{u}') \approx \mathbf{u}_h \cdot \boldsymbol{\nabla}(\mathbf{u}_h + \mathbf{u}')$ (see [74]).

ii) the subscales are not tracked in time, therefore, quasi-static subscales are considered here (see [87] for a justification of this choice); however, the subscale equation remains quasi time-dependent since it is driven by the large-scale time-dependent residual; (for time-tracking of subscales, see [88])

Substituting the approximated $\mathbf{u}'$ and p' into the large-scale equation and applying integration by parts we get:

$$\begin{cases} (\rho\partial_t \mathbf{u}_h, \mathbf{w}_h)_\Omega + (\rho\mathbf{u}_h \cdot \boldsymbol{\nabla}\mathbf{u}_h, \mathbf{w}_h)_\Omega - \sum_{K\in\mathcal{T}_h} (\tau_K \boldsymbol{\mathcal{R}}_{\mathrm{M}}, \rho\mathbf{u}_h \boldsymbol{\nabla}\mathbf{w}_h)_K + (2\mu\boldsymbol{\varepsilon}(\mathbf{u}_h) : \boldsymbol{\varepsilon}(\mathbf{w}_h))_\Omega \\ \qquad\qquad - (p_h, \boldsymbol{\nabla}\cdot\mathbf{w}_h)_\Omega + \sum_{K\in\mathcal{T}_h} (\tau_{\mathrm{C}}\mathcal{R}_{\mathrm{C}}, \boldsymbol{\nabla}\cdot\mathbf{w}_h)_K = (\mathbf{f}, \mathbf{w}_h)_\Omega \quad \forall \mathbf{w}_h \in V_{h,0} \\ (\boldsymbol{\nabla}\cdot\mathbf{u}_h, q_h)_\Omega - \sum_{K\in\mathcal{T}_h} (\tau_K \boldsymbol{\mathcal{R}}_{\mathrm{M}}, \boldsymbol{\nabla} q_h)_K = 0 \quad \forall q_h \in Q_h \end{cases} \tag{5.25}$$

In this work, we adopt the definition proposed in [89] for the stabilizing parameters:

$$\tau_{\mathrm{K}} = \left[\left(\frac{2\rho}{\Delta t}\right)^2 + \left(\frac{2\rho\|\mathbf{u_{h,K}}\|}{h_K}\right)^2 + \left(\frac{4\mu}{m_K h_K^2}\right)^2 \right]^{-1/2}, \tau_{\mathrm{C}} = \frac{h_K\|\mathbf{u_{h,K}}\|}{2} min\left(1, Re^h\right) \tag{5.26}$$

In the above definition Re^h is the local Reynolds number given by:

$$Re^h = \frac{\rho\|\mathbf{u_{h,K}}\|\mathbf{h_K}}{2\mu}$$

,and the coefficient m_K is a constant independent from h_K [90], h_K being the characteristic length of the element.

Note that the calculation of h_K is crucial in this work. Recall that the stability coefficients weight the extra terms added to the weak formulation (5.25) and they are defined for each element K of the triangulation (5.26). Typically, these coefficients depend on the local mesh size h_K. Many numerical experiments show that good results can be obtained when using the minimum edge lenght of K [91], while others use always the triangle diameter (see [92] for details).

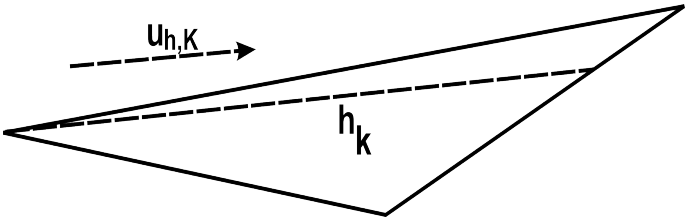


FIGURE 5.1: Longest triangle lenght in the streamline direction

However, in the case of strongly anisotropic meshes with highly stretched elements, the definition of h_K plays a critical role in the design of the stabilizing coefficients [93]. For advection dominated problem, the authors in ([89]) propose to compute h_K as the diameter of K in the direction of the velocity u as follows (see Figure 5.1):

$$h_K = \frac{2|\mathbf{u_{h,K}}|}{\Sigma_{i=1}^{N_K}|u_{h,K}\cdot\nabla\varphi_i|} \tag{5.27}$$

where N_K is the number of vertices of K and $\varphi_1, ..., \varphi_{N_K}$ are the usual basis functions of $P_1(K)$ mapped onto K. Whereas in diffusion dominated problem h_k is computed as follows:

$$h_k = \sqrt{\frac{\nu}{\frac{1}{\Delta t} + 2\nu \int_K \epsilon(b_k) : \epsilon(b_k) dK}}$$

where ν is the kinematic viscosity, Δt the time step and $\epsilon(b_k)$ the bubble function of the element K.

5.3 SCPG: Thermal solver

Equations (5.3), (5.9), (5.10) and (5.14) can be represented by a single scalar transient convection-diffusion-reaction equation which reads:

$$\partial_t \Phi + \mathbf{u} \cdot \boldsymbol{\nabla} \Phi + \boldsymbol{\nabla} \left(k \boldsymbol{\nabla} \Phi \right) + r\Phi = f \tag{5.28}$$

where Φ is the scalar variable, $\mathbf{u}$ the velocity vector, k the diffusion coefficient, r the reaction coefficient and f a source term. The solution strategy for solving such an equation is similar to that used for the equations of motion. Again, the spatial discretization is performed using approximation spaces. Thus, the Galerkin formulation is obtained by multiplying these equations by appropriate test functions, applying the divergence theorem to the diffusion terms and integrating over the domain of interest. Following the lines on the use of stabilization methods for transient convection-diffusion-reaction equations as discussed in [79, 94], the stabilized weak form of equation (5.28) reads:

$$\begin{cases} \text{Find } \Phi \in V_h \text{ such that, } \forall w \in V_{h,0} \\ (\partial_t \Phi + \mathbf{u} \cdot \boldsymbol{\nabla} \Phi, w) + (k \boldsymbol{\nabla} \Phi, \boldsymbol{\nabla} w) + (r\Phi, w) \\ \qquad + \underbrace{\sum_K \left(\mathcal{R}(\Phi), \tau_{\text{SUPG}} \mathbf{u} \cdot \boldsymbol{\nabla} w \right)_K}_{\text{streamline upwind}} + \underbrace{\sum_K \left(\mathcal{R}(\Phi), \tau_{\text{SCPG}} \tilde{\mathbf{u}} \cdot \boldsymbol{\nabla} w \right)_K}_{\text{discontinuity-capturing}} = (f, v) \end{cases} \tag{5.29}$$

where $\mathcal{R}(\Phi)$ is the appropriate residual of equation (5.28)); $\mathbf{u}$ is the convection velocity and $\tilde{\mathbf{u}}$ is an auxiliary vector function of the temperature gradient. In equation (5.29), two additional stabilizing terms have been introduced; the first controls the oscillations in the direction of the streamline (SUPG) [80, 95] and the other controls the derivatives in the direction of the solution gradient (SCPG) [96, 97]. This can improve the result for convection dominated problems while the shock-capturing technique precludes the presence of overshoots and undershoots by increasing the amount of numerical dissipation in the neighborhood of layers and sharp gradients. The evaluation of the τ_{SUPG} and τ_{SCPG} stabilizations are done following the definitions in [89, 90], and [98] respectively. Again these stabilization terms depend on the mesh size and thus, as for the VMS

solver, it must be adapted to anisotropic elements. Therefore the stabilization terms are computed as follows:

$$\tau_{\text{SUPG}} = \left(\left(\frac{1}{\Delta t} \right)^2 + \frac{2\|v\|_K}{h_K} + 9 \left(\frac{4k}{h_K^2} \right) \right)^{-\frac{1}{2}}$$

$$\tau_{\text{SCPG}} = \frac{h_K}{2|u_c|} \eta \left(\frac{|u_c|}{2|u|} \right)$$

with $\eta(\beta) = 2\beta(1-\beta)$, $\beta \in [0,1]$, and $|u_c| = \begin{cases} \frac{u.\nabla\Phi_h}{\nabla\Phi_h} & \text{if } \nabla\Phi_h \neq 0 \\ 0 & \text{otherwise} \end{cases}$

We used in the following numerical tests an implicit backward-Euler (implicit) time-integration scheme for equations (5.29) and (5.24). The algebraic problems resulting from the finite element formulation are assembled and solved using the conjugate residual method associated to the incomplete LU preconditioning from the PETSc (Portable Extensive Toolkit for Scientific Computation) library.

The CimLib library [99] is fully parallel involving the use of SPMD (Single Program, Multiple Data) modules and the MPI (Message Passing Interface) library standard [22]. All the steps are parallelized including the assembly of algebraic problems through PETSc as well as the partitioner and the meshing [24].

One last important feature of the proposed approach is that all the three-dimensional stabilized finite-element methods presented in this section, which are needed for solving the transient heat transfer and turbulent flows inside the furnaces, are completely suited with the Immersed Volume Method (presented in Chapter 1) approach without additional efforts.

5.4 Stabilized solvers and anisotropic mesh adaptation

Before testing the developed numerical methods with the anisotropic mesh adaptation, let us consider an example that validates the higher order of convergence obtained when using these tools. It was stated in [100] that when using stabilization methods, we loose half an order of convergence because of the added diffusion. However, together with the anisotropic adaptation, this loss is recovered and a global second order convergence is reached. To illustrate this point we take a convection dominated problem with boundary layer for which the exact solution is given by:

$$\Phi(x,y) = xy \left(1 - \exp\left(\frac{1-x}{a} \right) \right) \left(1 - \exp\left(\frac{1-y}{a} \right) \right)$$

This test has been studied by several authors [100, 101]. We consider the computation domain $\Omega = (0,1)^2$, the velocity field $\mathbf{u}(\mathbf{x},\mathbf{y}) = (\mathbf{1},\mathbf{1})^{\mathbf{T}}$ and varying diffusion coefficient $a = 10^{-1}, 10^{-3}, 10^{-6}$. The solution Φ develops boundary layers at $x = 1$ and $y = 1$. When the diffusion coefficient tends to 0, the flow becomes convection dominated and thus the standard Galerkin approach leads to the appearance of spurious oscillations. The latter are avoided and a smooth solution is obtained when applying the SUPG stabilization with anisotropic mesh adaptation. Recall that the amount of added artificial diffusion is related to the mesh size inside the layer region. This is computed as the largest edge of the element in the direction parallel to the velocity field. We can observe in Figure 5.2 that as the diffusion coefficient a tends to zero, the numerical solution becomes steeper without the appearance of any numerical oscillation. Figure 5.3 shows the anisotropic meshes made up of 20,000 elements obtained for the different values of the diffusion coefficient. Note the concentration of the resolution along the boundary layers. This reflects how, for a controlled number of nodes, the mesh is naturally and automatically coarsened in smooth regions while extremely refined near the boundary. The zoom on the right side of the cavity illustrates the sharp capture of the boundary layers and the right orientation and deformation of the mesh elements (longest edges parallel to the boundary). This yields a great reduction of the number of triangles and consequently a reduction in the computational cost.

This example aims at emphasizing the spatial order of convergence when using the proposed mesh adaptation technique. The global convergence order is computed in the L_∞, L_2 and H_1 norms by numerical integration. In each case, the error has been computed with respect to the reference solution. As can be seen in Figure 5.4 the anisotropic mesh adaptation proves to be very efficient in recovering the order of convergence of the method and even get twice higher convergence for the L_2 norm. Therefore the use of the anisotropic mesh adaptation method allows the recovery of the gobal convergence order of the numerical schemes while producing accurate and oscillation free numerical solutions. More test cases on this subject are treated in [102].

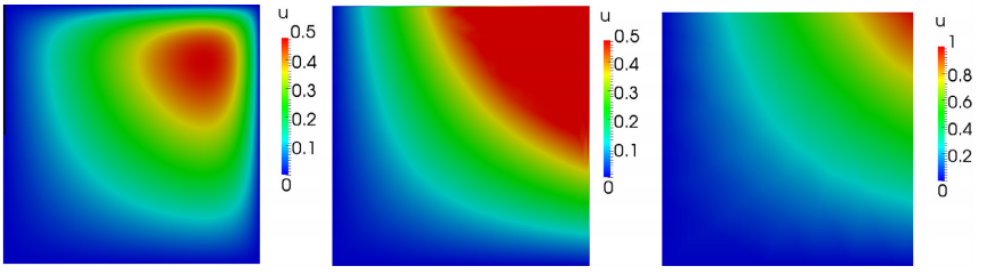


FIGURE 5.2: Numerical solution for $a = 10^{-1}, 10^{-3}, 10^{-6}$

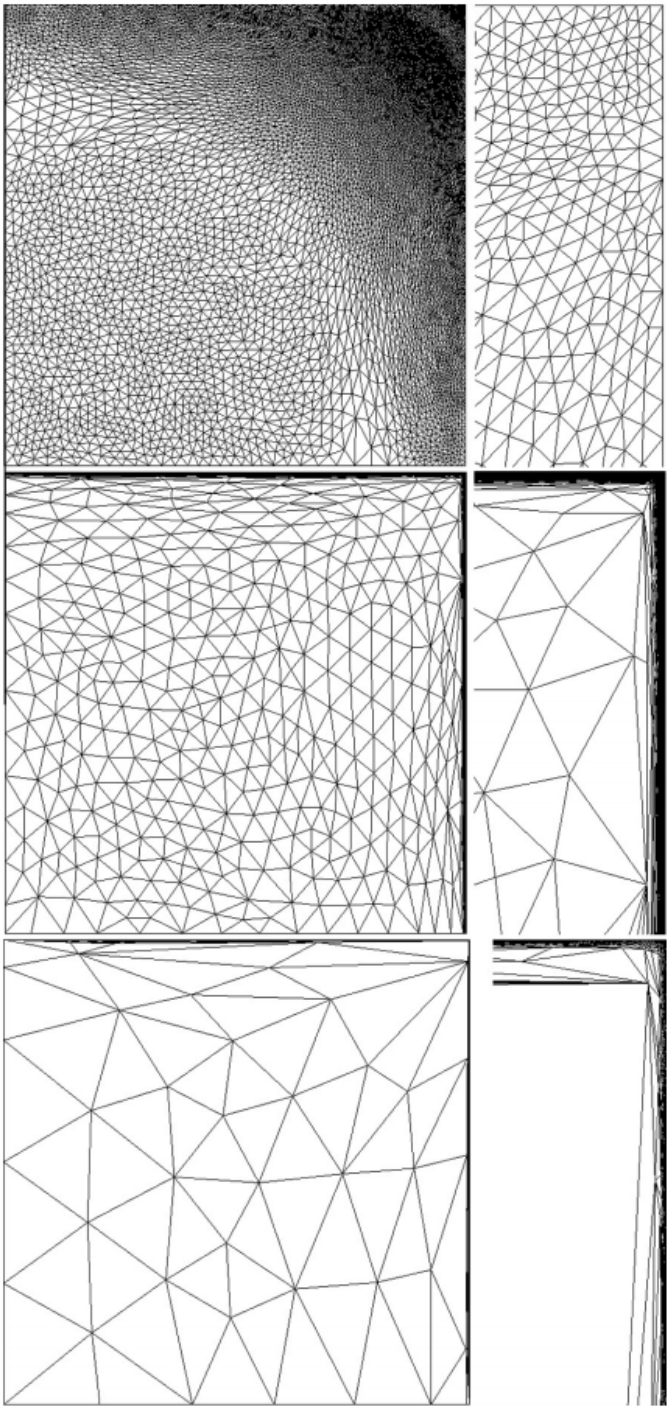


FIGURE 5.3: Anisotropic meshes for $a = 10^{-1}, 10^{-3}, 10^{-6}$

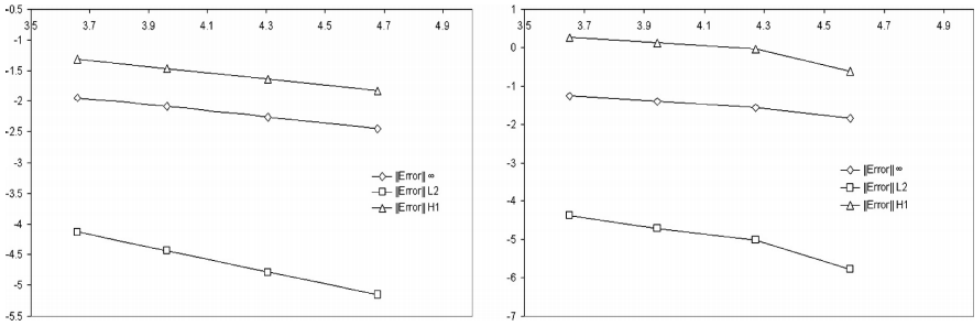


FIGURE 5.4: L_∞, L_2 and H_1 norms of the error versus the number of elements in the mesh for $a = 10^{-1}, 10^{-3}, 10^{-6}$

5.5 Numerical simulation of the heating of four ingots in a 2D furnace by forced convection

We consider a $2D$ problem of turbulent flow coupled to conjugated heat transfer (Figure 6.3). Four ingots are located in the cavity. The initial temperature of the cavity is set to $T_{ini} = 100^{\circ}C$ at the beginning and hot air is injected at the inlet at temperature of $T_h = 1300^{\circ}C$ and a velocity of $V_{in} = 10ms^{-1}$. These extreme conditions are close to what we have usually inside industrial furnaces. There are two outlets on the opposite side of the domain of height $0.5m$ each. The inlet is $1m$ wide. The dimensions and physical properties of the four ingots and the cavity are listed in the table 6.1. The fields given to the edge-based error estimator are the temperature, the velocity vector and the level-sets of the four ingots. The anisotropic mesh adaptation (presented in the next chapter) has been used. The mesh is adapted on the temperature, the velocity and the level-set functions of the four solids and is made of around $15,000$ nodes.

Figure 5.5 shows the streamlines in the cavity at different times. A main flow can be visualized from the inlet to the outlets. Several vortices are localized between the solids, but also in the corners of the cavity. They stay almost at the same place during the computation. The temperature is also plotted on the fluid-solid interface. The interface corresponds to the zero-isovalue of the level-set. One can notice the good definition of the latter. In fact the mesh is dense all along the solid interface as it has been used as a mesh adaptation criterion. Moreover the temperature gradients are localized at the interface (Figures 5.6 and 5.7). During the simulation the solids are heated by the injected air. We observe that a faster raise in temperature is obtained on the left ingot. This was expected as this ingot is the first facing the jet and as it has the lowest density. Obviously the density is a key parameter for the temperature evolution. The right ingot is placed far from the inlet and close to the outlets, but as its density is higher than the middle right one, it is heated faster.

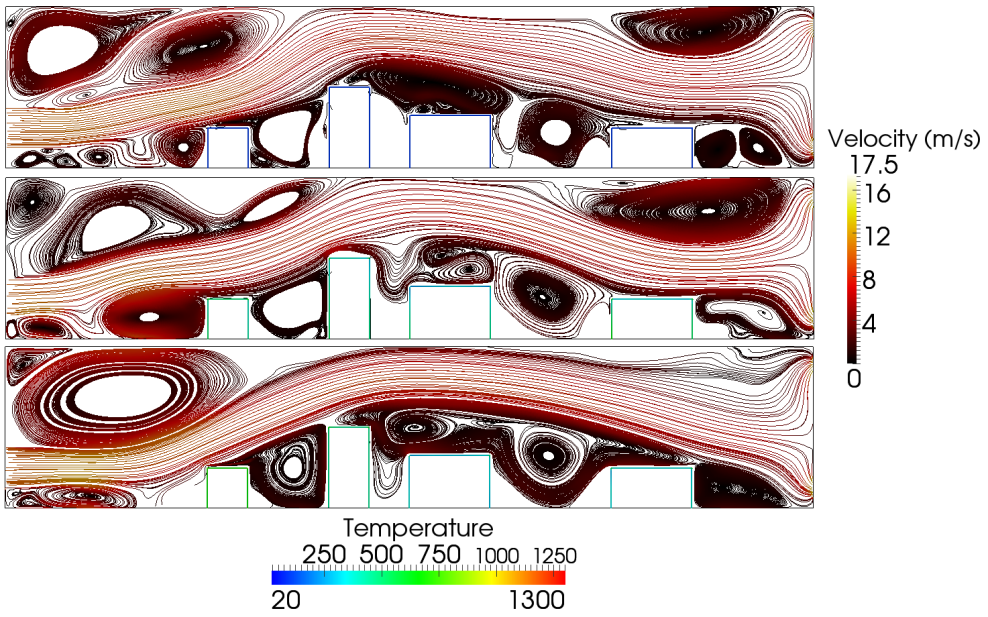


FIGURE 5.5: Streamlines in the cavity and temperature at the solid-air interface at different times

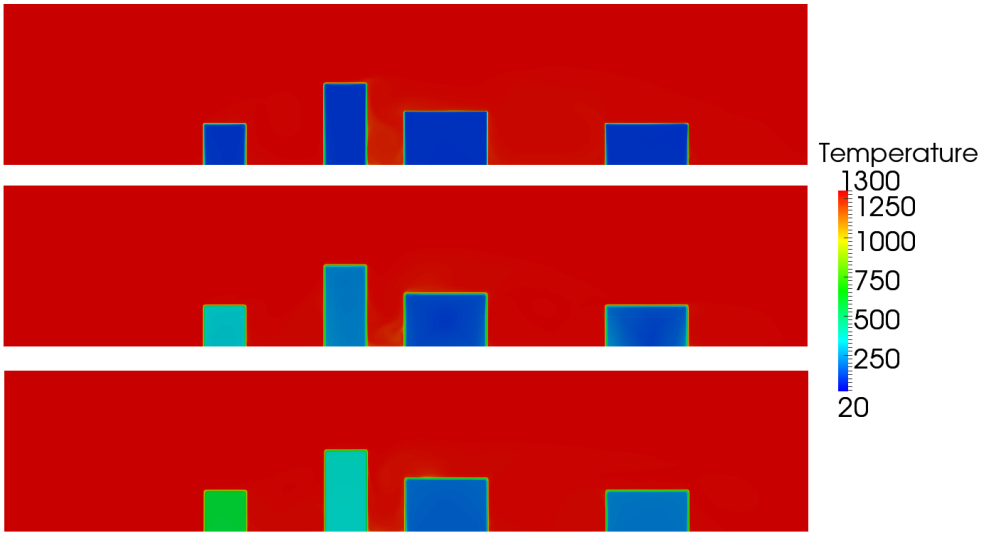


FIGURE 5.6: Temperature distribution in the cavity at different times

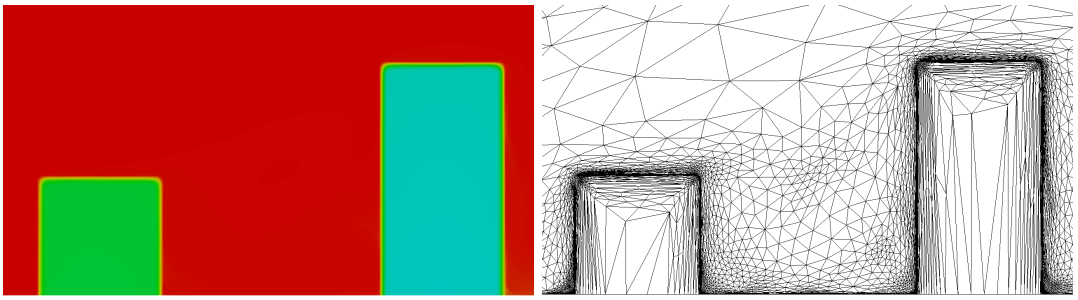


FIGURE 5.7: Zoom on the temperature distribution at the interfaces (top), and the corresponding mesh (right)

This case is a good demonstration of the capability of the solvers coupled to the anisotropic mesh adaptation to solve turbulent flows with high temperature gradients (we remind that here hot air at temperature $1300^{\circ}C$ is directly injected in a cavity at ambiant temperature). It shows the flexibility of the mesh adaptation method to follow several fields at the same time and the capacity of the solvers to provide a stable solution over highly streteched elements.

5.6 3D numerical simulation of an industrial furnace

In this section, we aim to present 12 hours of the heating process of an industrial furnace given by an industrial partner. Figure 5.8 shows six ingots with arbitrary geometries taken initially at $400^{\circ}C$ and positioned at different locations inside the furnace. All the computations have been conducted by starting with a gas at rest and at a constant temperature of $700^{\circ}C$. At all other boundaries, a constant flux of $400W/m^2$ is applied for sake of simplicity. The air is vented out of the furnace through two outlets positioned at the bottom vertical wall. An adaptive time-step is used starting from $0.001s$ and increases as the solution stabilizes. The 3D computations have been obtained using 40 $2.4Ghz$ Opteron cores. We can identify two types of ingots; thick placed on the left wall (1, 2 and 4) and thin placed on the right wall (3, 5 and 6).

The furnace is modelled as a hexagonal section duct of 2.7 x 8.1 x 5.3 m^3 forming one heat transfer zone. The hot gas is pumped into the furnace through one burner located on the vertical wall having a constant speed of $38m/s$ and temperature of $1350^{\circ}C$. We can clearly see the burner in a real furnace in Figure 5.9 and how we insert the workpieces from the opened top hatch. For more details about the geometry, we present in Figure 5.10 the CAD from different angle views of the furnace.

By applying the IVM method, the levelset function first detects and defines the treated objects. The second step consists of deriving the anisotropic adapted mesh that describes very accurately the interface between the workpieces and the surrounding air. Recall

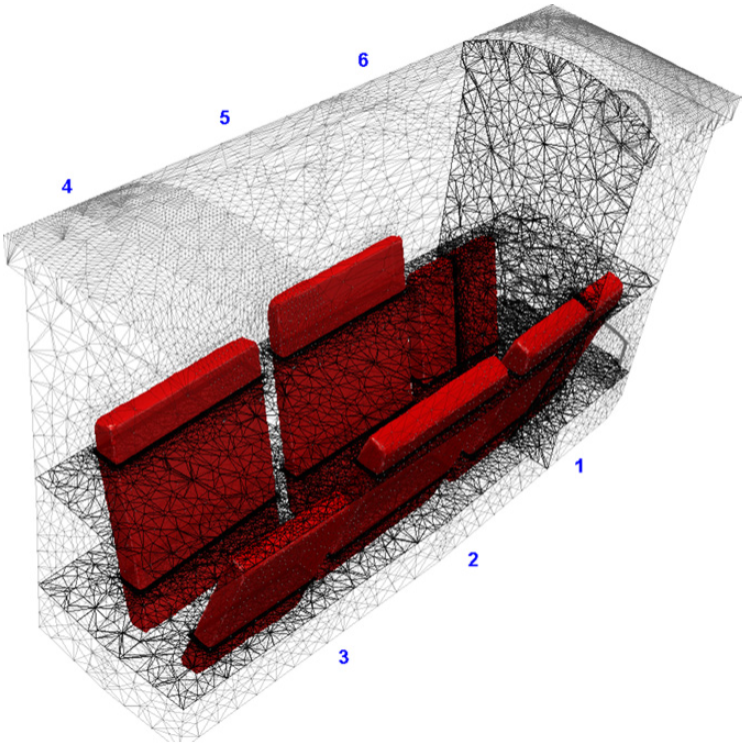


FIGURE 5.8: Computational domain after anisotropic mesh adaptation.

FIGURE 5.9: A top view of the furnace and the immersion of an ingot inside the furnace

that the mesh algorithm allows the creation of extremely stretched elements along the interface, which is an important requirement for multi-material problems with surface conductive layers. The additional nodes are added only at the interface region keeping the computational cost low.

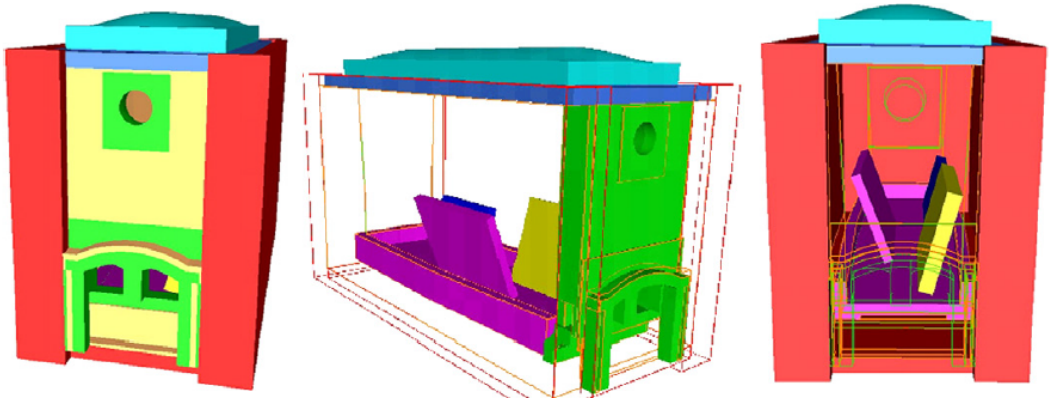


FIGURE 5.10: Different view angles of the furnace.

The algorithm progressively detects and refines the mesh at the fluid-solid interfaces leading to a well respected shape in terms of curvature, angles, etc. All the small details in this given geometry can be captured accurately (see Figure 5.8). Note that the final mesh used for the numerical simulation consists of $157,347$ nodes and $884,941$ tetrahedral elements.

Once the mesh is well adapted along the interfaces, the material distribution between the physical domains can be described by means of the level set function. Consequently, the same set of equations; momentum equation, energy equation, the turbulent kinetic, dissipation energy equations, and radiative transport equation are simultaneously solved over the entire domain including both fluid and solid regions with variable material properties (see Table 5.1). In the numerical simulation, the heat capacity C_p, the conductivity

TABLE 5.1: Properties of materials.

Properties	Smoke	Steel 40CDVL3
density ρ [kg/m^3]	1.25	7,800
heat capacity C_p [J/(kg K)]	1000	600
viscosity μ [kg/(m s)]	1.9e-5	–
conductivity λ [W/(m K)]	0.0262	37
emissivity ϵ	–	0.87

λ and the emissivity ϵ of the smoke and the steel are thermo-dependent. The emissivity of the smoke was computed from the proportions of the H_2O and CO_2 issued from the combustion, the thickness of the smoke and the temperature as in the model studied in [103].

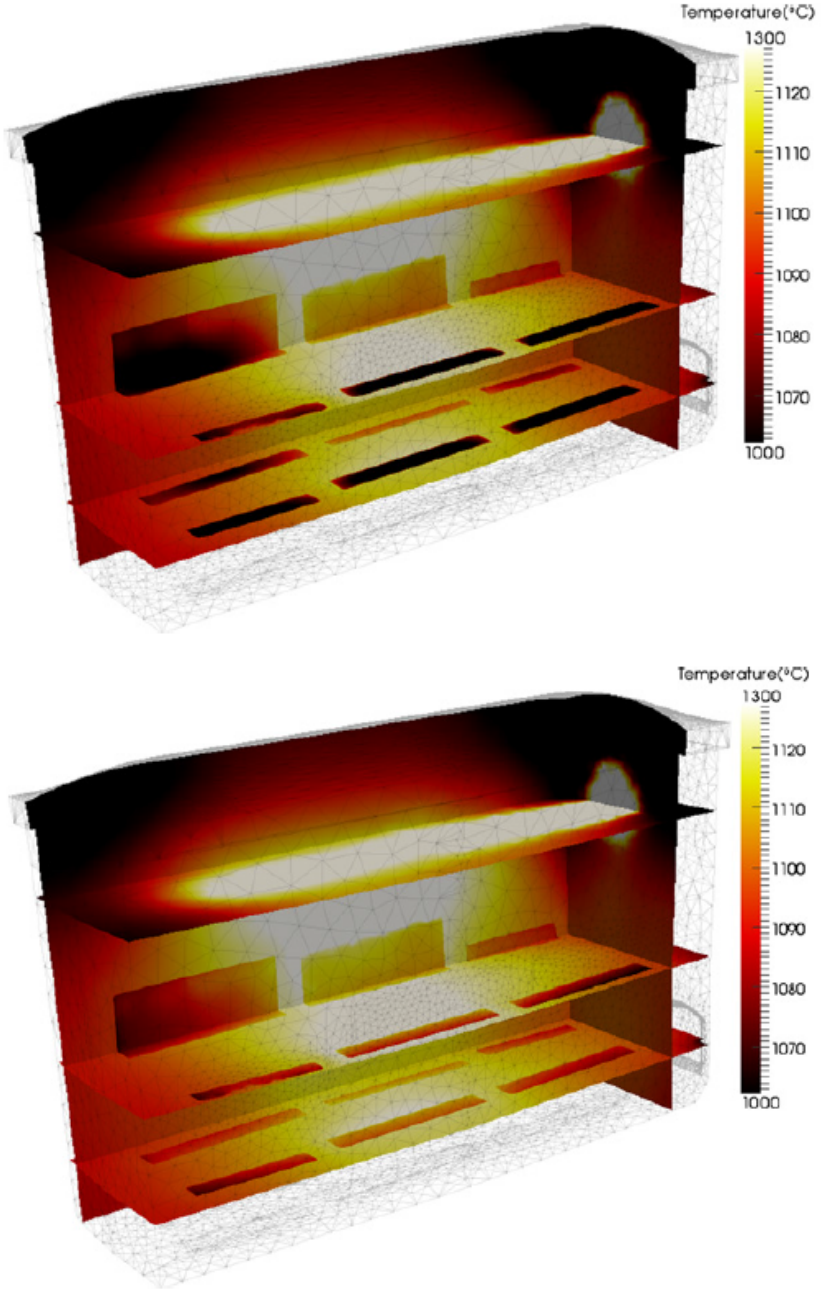


FIGURE 5.11: Streamlines and isotherms inside the furnace at two different time steps.

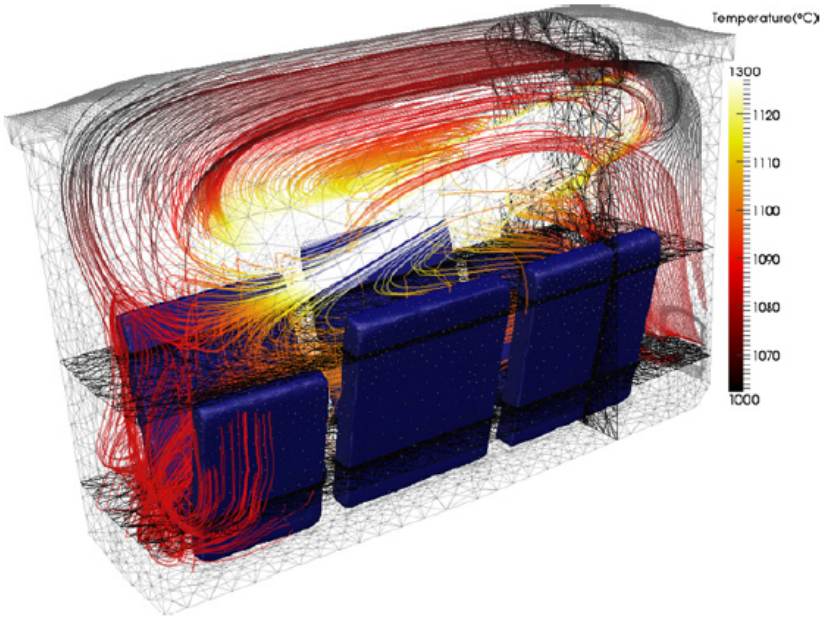


FIGURE 5.12: Streamlines distribution inside the furnace and around the ingots.

All the given parameters used for the numerical simulations do not reflect the true measurements from the experimental tests, due to the complexity of the wall properties, the gas composition and other technical issues. However, we made sure that the chosen parameters have at least the real physical representations and are appropriate to simulate the real test.

Figure 5.11 shows the temperature distribution on four mutually parallel planes in the furnace for two different times (t= 1.25s and 222s). The temperature distribution clearly indicates the expected flow pattern. At the solids level, we observe that the injected air from the top burner is slowed down and slightly influences the main air circulation in this part of the domain. This explains the difference in the flow pattern between the top and bottom part of the furnace. When the hot fluid passes across the volume of the furnace, it induces a turbulent and recirculating motion within the geometry. This forced convection is caused by the interaction of the moving stream and the stationary fluid inside the furnace. The air movement around the workpieces is quite complex and interesting; i.e. it allows the study of the influence of different arrangements and positions to optimize the heat treatment. A number of vortices between the objects and the surroundings can be observed due to the turbulence dissipation and mixing between the hot and cold air. All these observations are highlighted by the streamlines in Figure 5.12 and the velocity components in Figures 5.13 and 5.14.

Moreover, we can clearly see on these vertical planes cutting through the ingots that the solid region satisfies the zero velocity and, hence, the no-slip condition on the extremely refined interface is also verified. The obstacles (6 ingots) slow down the air circulation

in the bottom zone of the furnace and slightly influence the main air circulation along the walls.

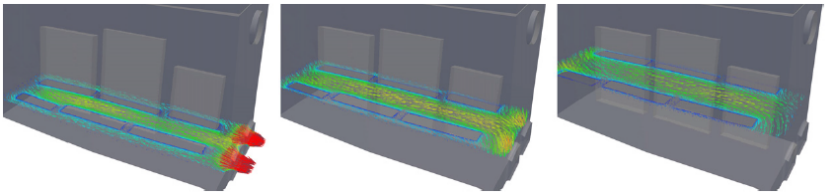


FIGURE 5.13: Velocity vectors on different cut-planes inside the furnace.

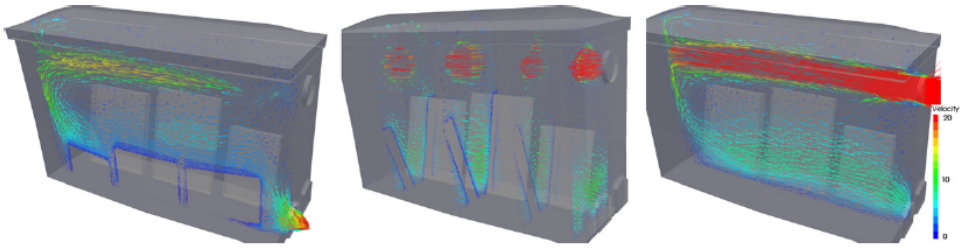


FIGURE 5.14: Velocity vectors on different cut-planes inside the furnace.

To get better information on the time history of the temperature, we plot in Figure 5.15 the evolution captured at the center of the ingots. As expected, we notice that the thin ingots (3, 5 and 6) in general are heated faster than the thick ones (1, 2 and 4). At the same time, the temperature of the ingots positioned in the center and facing the flame jet continuously, increases faster than the others. This is due to the fact that the flames hit the walls and deviate towards the center forming a slight counter clockwise rotating flow. Near the center of the furnace and under the flame jet, a full rotating gas flow is always present, which is ended near the impeller bottom-surface and exits through the two outlets.

One can also observe in Figure 5.15 the presence of a certain phase change in the material properties. The favorable and the reasonable nature of such results showed a good potential for the developed formulations. However, comparisons with experimental results having true workpieces geometry and positioning will be the subject of further investigations.

It is also worth mentioning that the profiles of the temperature do not suffer from spurious oscillations (undershoots or overshoots) which are frequently observed in the presence of high temperature gradients at the interface or in convection dominated problems across the enclosure. This can be attributed to the stabilized finite element discretization applied on the system of equations (5.4)-(5.6). Summing up, the combination of the local mesh adaptation and the use of iterative solvers together with the smoothed distribution of the thermo-physical properties across the interface overcome the numerical instabilities and lead to good numerical behavior.

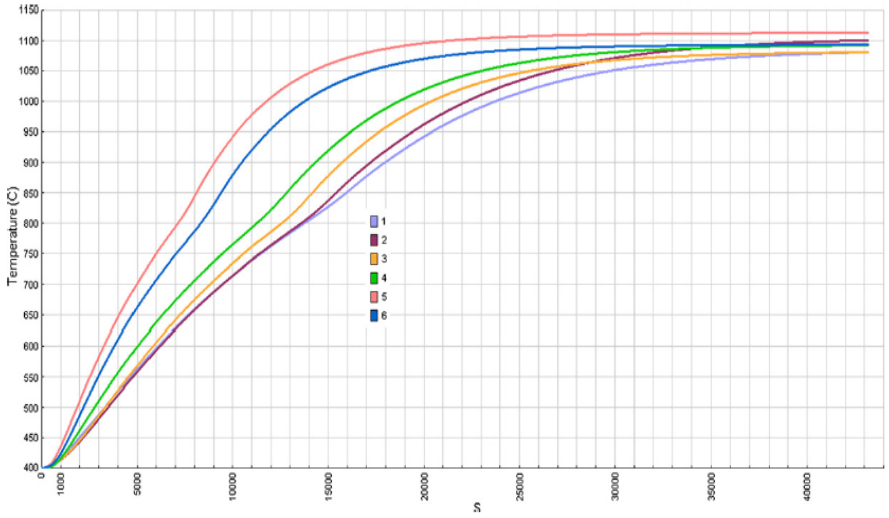


FIGURE 5.15: Temperature-time profile at different locations in the furnace

These numerical results indicate that the Immersed Volume Method (presented in Chapter 1) approach is suitable for the numerical simulation of industrial furnaces with different loads. Such calculations allow the prediction of different parameters and the understanding of the flow characteristics for heat treatment furnaces. Future investigations must be concerned with experimental comparisons and reducing simulation time for industrial models.

5.7 3D numerical simulation of an industrial furnace with dynamic anisotropic mesh adaptation

We consider the case presented in the previous section and added the dynamic anisotropic mesh adaptation method. The objective is to validate the methods (Immersed Volume Method, stabilized solver and anisotropic mesh adaptation) on a complex 3D industrial case which is highly turbulent and facing high temperature gradients.

The six ingots are positioned at the same places, but their initial temperature is $23^{\circ}C$. The furnace is initially at temperature $640^{\circ}C$. Hot air at temperature $1352^{\circ}C$ is injected through the inlet with a velocity of $38ms^{-1}$ and is evacuated through the two outlets at the bottom of the furnace. In order to analyze the performance of the anisotropic mesh adaptation method, we have run three cases. The first case has been run with a dynamic mesh adaptation. The adaptation takes into account at the same time the temperature, the velocity and the level-set functions of the six ingots. The mesh is composed of about $500,000$ elements. The second and the third cases have been run using fixed uniform meshes. The mesh of the second case is formed of $250,000$ elements and the mesh of the

third has about 500,000. In what follows we analyze and compare the results obtained with these three simulations.

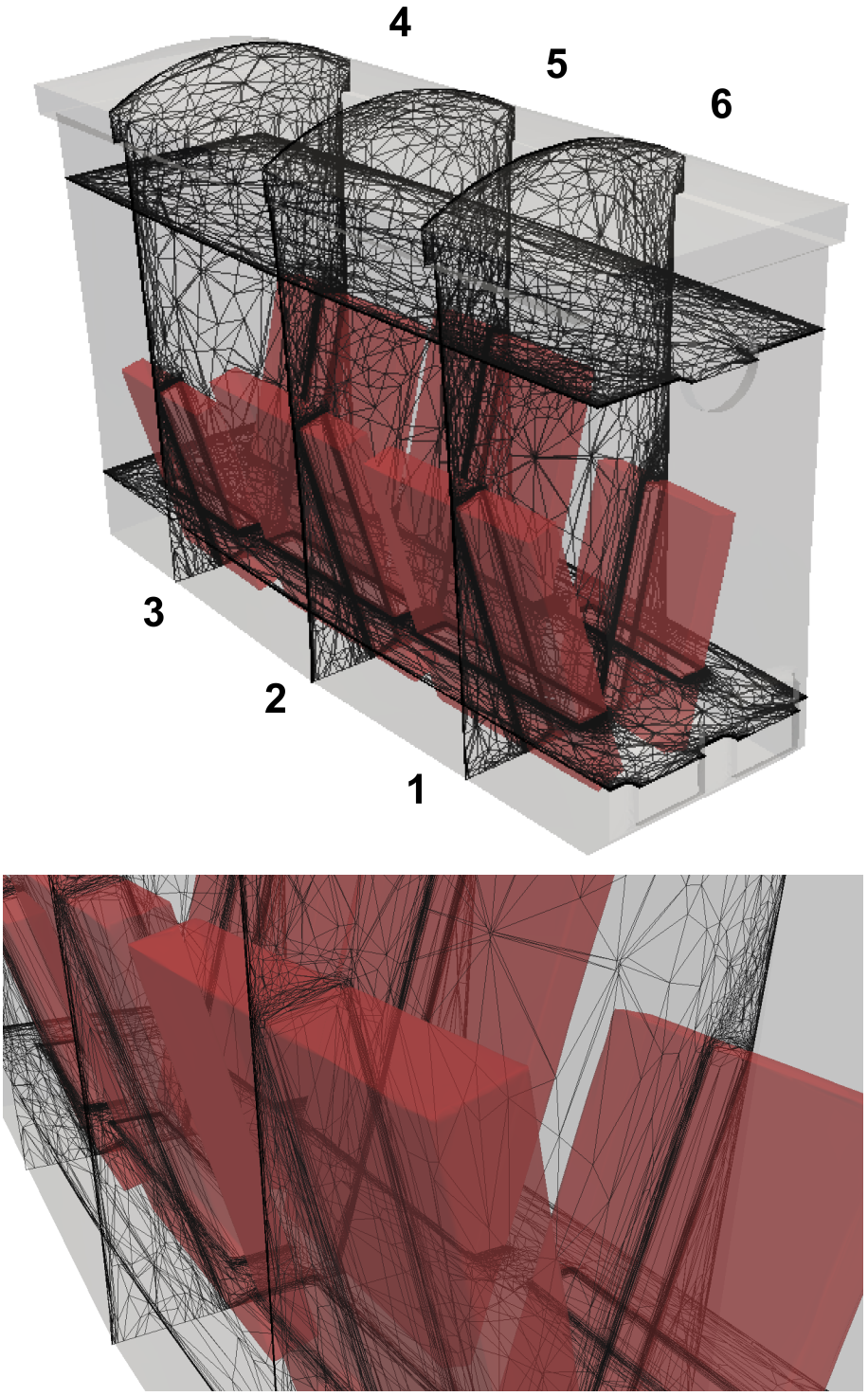


FIGURE 5.16: Cuts of the mesh in the furnace and the ingots numbering (top). A zoom on the adapted mesh at the boundaries and the interfaces (bottom)

Figure 5.16 shows the ingots position in the furnace as well as their numbering. We remind that ingots 1, 2 and 4 are thicker than ingots 3, 5 and 6. One can also notice

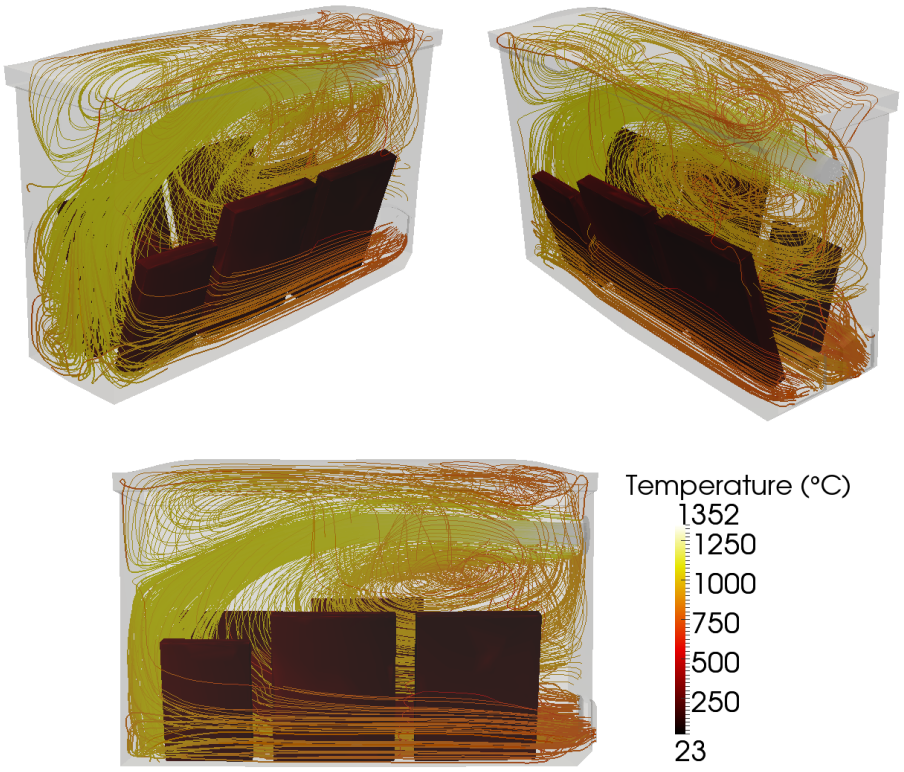


FIGURE 5.17: Streamlines inside the furnace

in Figure 5.16 that the interface is captured better (especially the edges) using the anisotropic mesh adaptation than the case in the previous section (Figure 5.8). Figure 5.17 shows the streamlines in the furnace. The main flow blows out from the inlet, hits the opposite wall, goes down and get out through the two outlets. This main flow creates vortices in the upper part and at the center of the furnace. Figure 5.18 presents cuts for the velocity and the temperature. One can see the heated air coming at a higher speed from the inlet, the progressive heat of the ingots and the obtained boundaly layers at the walls and around the ingots.

Sensors have been placed in the ingots during the simulation and the corresponding curves are provided in Figures 5.19 and 5.20. We can see that the thin ingots (3, 5 and 6) are heated faster than the thick ones (1, 2 and 4). The position along the furnace is also important. The ingots close to the outlets are heated slower than the others. These results are quatitavely in good agreement with the ones obtained with the static mesh simulation in the previous section. The curves also provide a comparison between the case run with mesh adaptation and the cases run with a fixed mesh. The ingots of the fixed mesh simulations are heated faster at the beginning of the simulation but then the rise in temperature is slower compared to the mesh adaption case. But as the number of elements is increased ($250,000$ to $500,000$) the results get closer to the adapted case. Hence we can conclude that the anisotropic mesh adaptation provides a better accuracy.

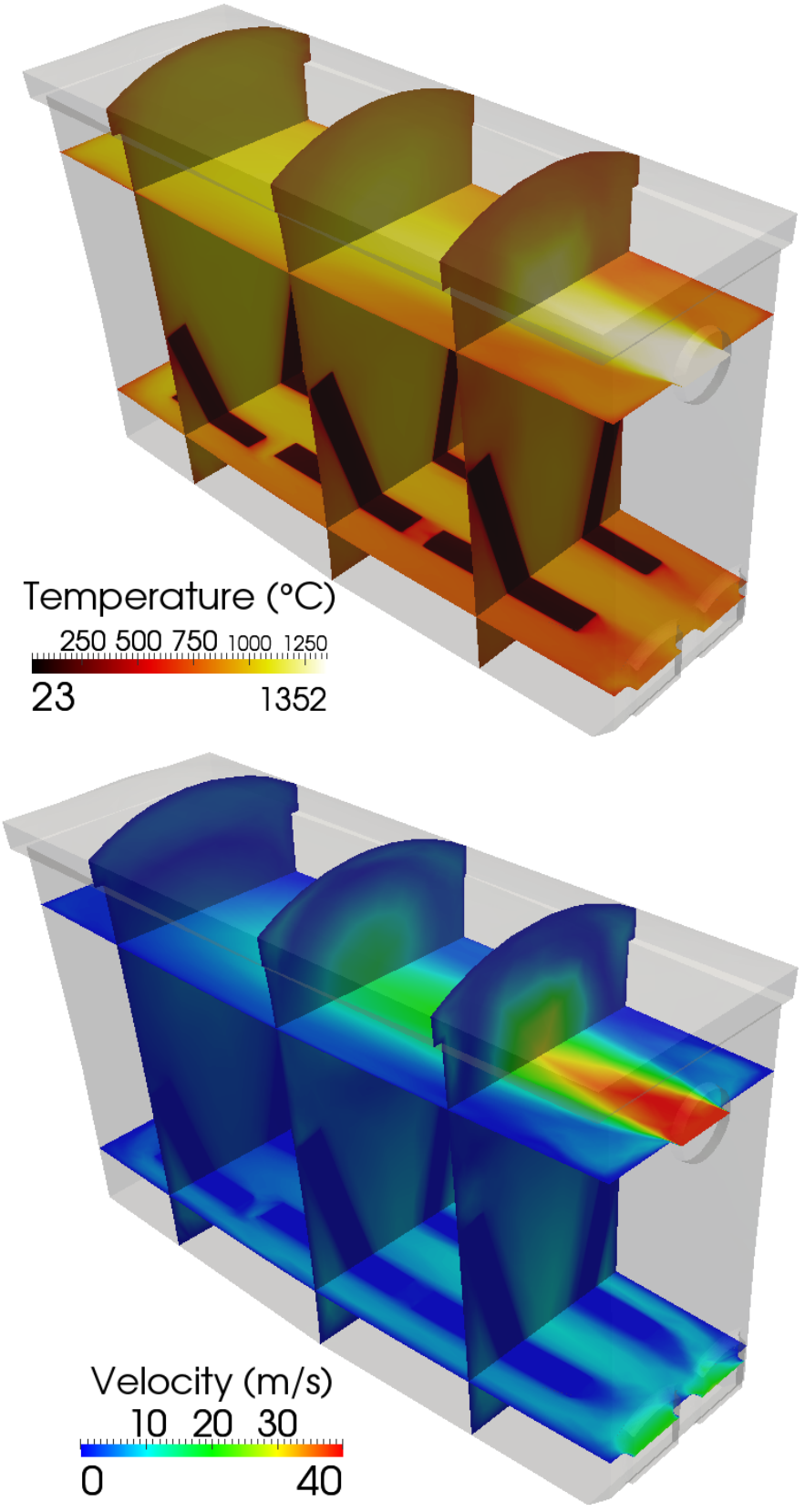


FIGURE 5.18: Cuts of the temperature (top) and the velocity (bottom) in the furnace

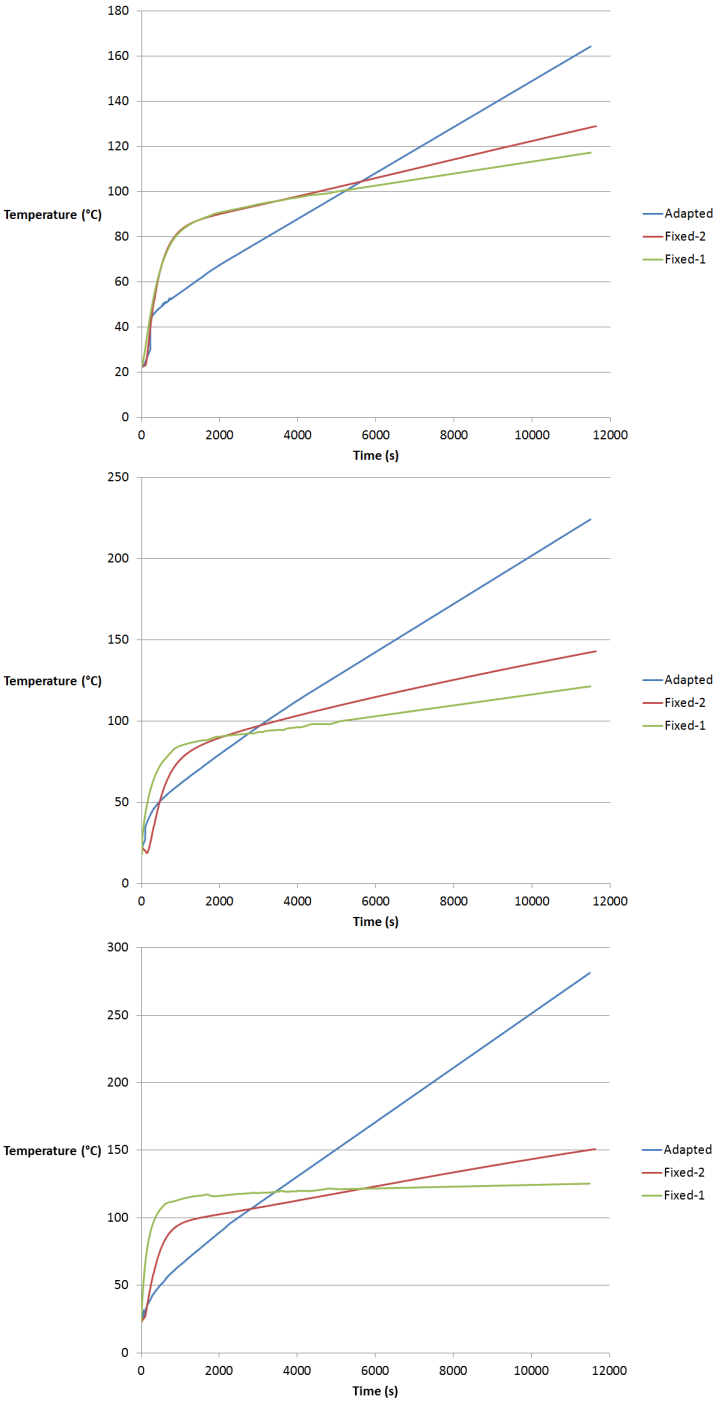


FIGURE 5.19: Temperature evolution of ingots 1 (top), 2 (middle) and 3 (bottom) in the three cases. The curve "Fixed-1" corresponds to the fixed mesh of $250,000$ elements, "Fixed-2" to the fixed mesh of $500,000$ elements and "Adapted" to the dynamic anisotropic adapted mesh case.

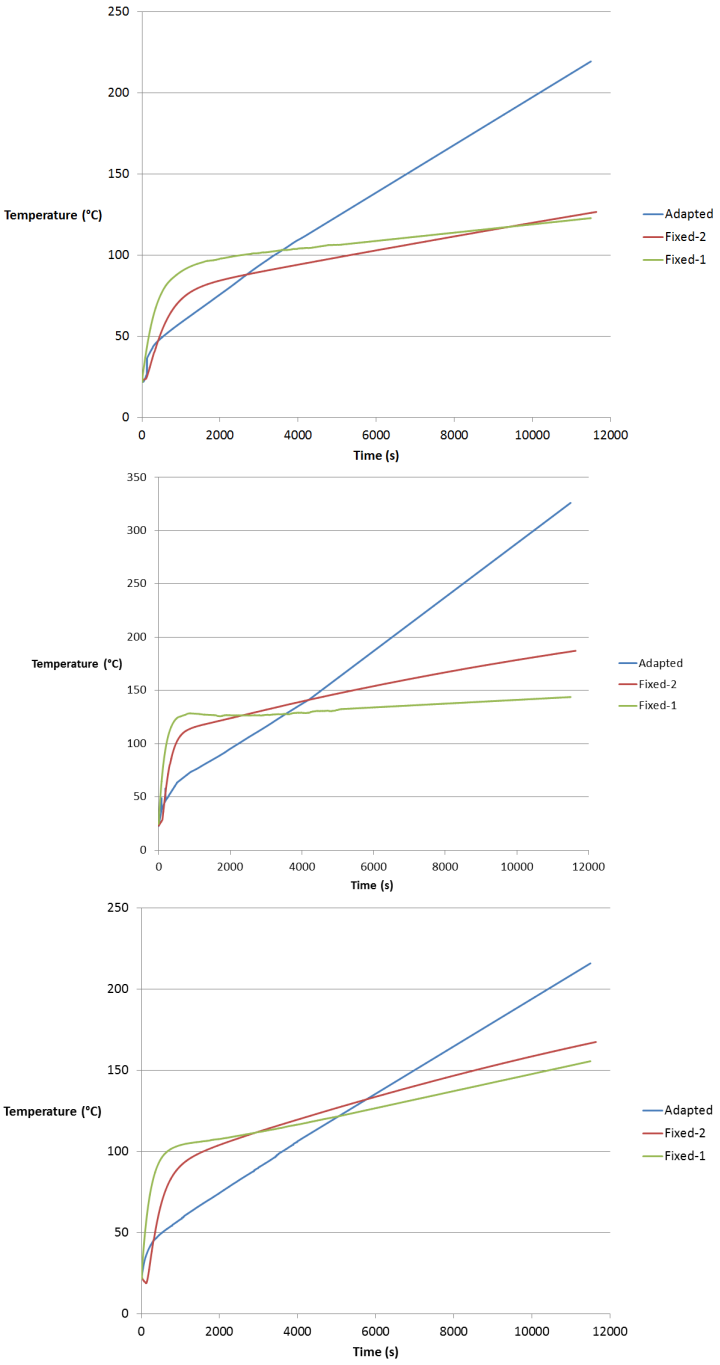


FIGURE 5.20: Temperature evolution of ingots 4 (top), 5 (middle) and 6 (bottom) in the three cases. The curve "Fixed-1" corresponds to the fixed mesh of $250,000$ elements, "Fixed-2" to the fixed mesh of $500,000$ elements and "Adapted" to the dynamic anisotropic adapted mesh case.

Table 5.2 shows the computational time in hours for each of the three cases. We can see that the adapted case is clearly slower as its computational time is more than 60% higher than the fixed-mesh case with the same number of elements $(500,000)$. This extra computational time corresponds to the mesh modification operations and forms an interesting perspective subject.

TABLE 5.2: Computational time in hours of the three cases for $10,800s$

Adapted	Fixed-1	Fixed-2
476	162	292

5.8 Conclusion

In this chapter we have presented the numerical method used to solve turbulent flows and conjugated heat transfer problems. Stabilized schemes are used to overcome spurious numerical oscillations appearing for convection-dominated problems. The VMS method consists in decomposing both the velocity and the pressure fields into coarse/resolved scales and fine/unresolved scales. The fine-scale solution is substituted into the large-scale problem providing additional terms, tuned by a local stabilizing parameter, that enhance the stability and accuracy of the standard Galerkin formulation. The stabilization of the heat equation is done through the use of a SCPG scheme.

The use of anisotropic mesh adaptation is definitly justified because it allows to recover half an order of convergence because of the added diffusion. This point has been illustrated through the resolution of a convection-dominated case with a known solution. The gobal convergence order of the numerical schemes has been recovered while producing accurate and oscillation free numerical solutions.

The stabilized solvers as well as the anisotropic mesh adaptation have been tested on different 2D and 3D cases. This example have demonstrated the ability of the solvers to deal with turbulent flows and high temperature gradients on extremely stretched elements. Moreover the chosen strategy is capable of modeling industrial processes like the heating in industrial furnaces. The use of anisotropic mesh adpatation seem to provide more accurate results. The details of the anisotropic mesh adaptation method are provided in the next chapter.

Résumé français

Dans ce chapitre sont présentés les équations régissant les problèmes d'écoulements turbulents couplés aux flux thermiques. La mécanique des fluide est prise en compte par la résolution des équations de Navier-Stokes et la thermique par l'équation de la chaleur. Un modèle de turbulence $k - \epsilon$ est également utilisé pour certaines applications. Un modéle de radiation de type $P1$ est pris en compte pour inclure le flux radiatif dans l'équation de la chaleur.

Ces équations sont résolues par des méthodes éléments finis stabilisés afin d'éviter les oscillations numériques pouvant apparaître lors de problèmes à convection dominante. Le schéma temporel utilisé est de type implicite. La méthode stabilisée multi-échelle pour résoudre les équations de Navier-Stokes consiste à décomposer les champs de vitesse et pression en petites et grandes échelles. La solution des petites échelles est introduite dans les grandes échelles en vue de la résolution. Ainsi des termes de stabilisation sont ajoutés par rapport à la méthode standard Galerkin. Ces termes dépendant de la taille de maille, il est essentiel dans notre cas d'effectuer un calcul de cette dernière adapté à des éléments anisotropes. La partie thermique est stabilisée par la méthode SCPG.

L'utilisation de maillages anisotropes est justifiée une nouvelle fois. En effet les méthodes de stabilisation procurent la perte d'un demi ordre de convergence à cause de la diffusion ajoutée. La résolution sur des maillages anisotropes adaptés permet de récupérer ce demi ordre et ainsi d'optenir une convergence d'ordre 2. Cette propriété est vérifiée dans un exemple 2D où on adapte le maillage avec la méthode présentée dans le chapitre précédent sur un cas de convection dominante avec une solution connue.

Ensuite l'utilisation des solveurs stabilisés couplés à la méthode d'adaptation de maillage anisotrope est mise à l'épreuve sur plusieurs cas. Le premier est le cas de convection forcée 2D du chapitre précédent. La robustesse des solveurs à calculer une solution stable est également démontrée sur un cas de four industriel 3D. L'utilisation d'adaptation de maillage sur ce cas semble produire des résultats plus précis. Cette tendance sera vérifiée dans le prochain chapitre en comparant les résultats numériques avec des résultats expérimentaux.

Chapter 6

Anisotropic Mesh Adaptation Method

We remind that the target of this work is to reduce the time for computing problems with turbulent flows and conjugated heat transfer inside large domains. Therefore the use of uniform meshes is clearly unaffordable. Moreover, using anisotropic mesh adaptation is obvious since we deal with problems where the physics is highly anisotropic itself. Such an adaptation is crucial to capture all the physics at different scales of the problem. The objective is to construct an anisotropic adaptive mesh that minimizes the interpolation error over the computational domain. It is very well know in the litterature that the mesh conforming a metric is a unit mesh [29, 34, 104]. That is a mesh that transforms the edges of the domain into edges of unit length. However building a unit mesh in the Riemanian space results in a fine mesh everywhere and consequently yields to drastic computational costs. Therefore the idea is to build a unit mesh in the metric space. The resulting mesh will be anisotropic in the Riemanian space. The objective of this work is to build up a metric tensor on each node $\mathbf{X^i}$ of the mesh and to transform all the edges connected to $\mathbf{X^i}$ into unity:

$$\widetilde{\mathbf{X^{ij}}} = \mathbf{s_{ij}} \mathbf{X^{ij}}$$

In order to do that the stretching factors s_{ij} will be determined in terms of the interpolation error. A new edge based error estimator is proposed to evaluate the error on the edges. In the following section we will discuss how to evaluate this error using a new edge based error estimator.

The anisotropic adaptation is performed by constructing a metric map that allows the mesh size to be imposed by the variation of the gradient of any field (distance function, velocity, temerature...). We introduce first the so-called metric which is a symmetric

positive defined tensor representing a local base that modifies the distance computation [28–31], such that:

$$||\mathbf{x}||_{\mathrm{M}} = \sqrt{{}^t\mathbf{x} \cdot \mathbb{M} \cdot \mathbf{x}}\ , \quad < \mathbf{x}, \mathbf{y} >_{\mathrm{M}} = {}^t\mathbf{x} \cdot \mathbb{M} \cdot \mathbf{y}\ . \tag{6.1}$$

The metric $\mathbb{M}$ can be regarded as a tensor whose eigenvalues are related to the mesh sizes, and whose eigenvectors define the directions for which these sizes are applied. For instance, using the identity tensor, one recovers the usual distances and directions of the Euclidean space.

6.1 state of the art

Anisotropic mesh adaptation was first proposed in the late 1980s [105–108]. Significant research effort has been devoted in the last few years to develop successful anisotropic mesh adaptation techniques with real applications. Several approaches to easily build unstructured anisotropic adaptive meshes are often based on local modifications ([109], [110], [111], [30]) of an existing mesh. In fact, it mainly requires extending the way to measure lengths following the space directions and that can be done using a metric field to redefine the geometric distances. In parallel, theories on anisotropic a posteriori error estimation (*i.e.* [112]) have been well developed, leading to some standardization of the adaptation process; production of metrics from the error analysis of the discretization error and steering of remeshing by these metrics.

To resume, we distinguish four major error estimates for anisotropic adaptation: the hessian based relying on the solution hessian information to evaluate the linear interpolation error [29, 31, 113, 114], the a posteriori estimates approximating the discretization error using a theoretical analysis [115–119], the a priori error estimates [112, 120] and the goal oriented estimates that provide mathematical framework for assessing the quality of some functionals [104, 121–123]. Thanks to these technical and theoretical advances, a considerable improvement was obtained in the accuracy and the efficiency of numerical simulations.

Indeed, anisotropic mesh adaptation has proved nowadays to be a powerful strategy to improve the quality and efficiency of finite element/volume methods. It enables to capture scale heterogeneities that can appear in numerous physical or mechanical applications including those having boundary layers, shock waves, edge singularities and moving interfaces [29, 34, 124, 125]. In these cases, discontinuities or gradients of the solution are highly directional and can be captured with a good accuracy using an anisotropic mesh with stretched elements.

6.2 Edge-based Metric

In this section, the steps that constitute the computation of the metric for the anisotropic mesh adaptation will be outlined. The generation of the metric may be divided into consecutive steps. The main ones are the definition of a length distribution tensor followed by a gradient recovery procedure allowing the computation of an edge based error estimator. Finally a set of stretching factors associated to each edge may be computed, from which it is possible to define a new anisotropic metric.

6.2.1 Definition of the length distribution tensor: a statistical representation

In the computation of a discrete metric, one is faced with the problem of transferring element-wise information to the nodes. Rather than using a simple average, the process presented here takes into account a statistical representation of the lengths distribution of all the edges sharing a node. In order to define such quantity, let us consider a finite element discretization of the domain $\Omega = \bigcup_{K\in\mathcal{K}} K$ where K is a simplex such as a triangle or tetrahedron and $\mathcal{K}$ is the spatial discretization of the domain. We seek the solution in the space $\mathcal{V} = \mathcal{C}^2(\Omega)$. Let $\mathcal{V}_h$ be a simple P^1 finite element approximation space:$\mathcal{V}_h = \left\{w_h \in \mathcal{C}^0(\Omega), w_h|_K \in P^1(K), K \in \mathcal{K}\right\}$ and let us consider a function $u \in \mathcal{V}$. We define $\mathbf{X} = \left\{\mathbf{X}^i \in \mathbb{R}^d,\, i = 1, \cdots, N\right\}$ as the set of nodes in the mesh and we denote by U^i the nodal value of u at $\mathbf{X}^i$, finally let Π_h be the Lagrange interpolation operator from $\mathcal{V}$ to $\mathcal{V}_h$ such that: $\Pi_h u(\mathbf{X}^i) = u(\mathbf{X}^i) = U^i\,, \forall i = 1, \cdots, N$. As shown in Figure 6.1, $\Gamma(i) = \left\{j\,, \exists K \in \mathcal{K}\,, \mathbf{X}^i, \mathbf{X}^j \text{ are nodes of } K\right\}$ denotes the set of nodes connected to node i .

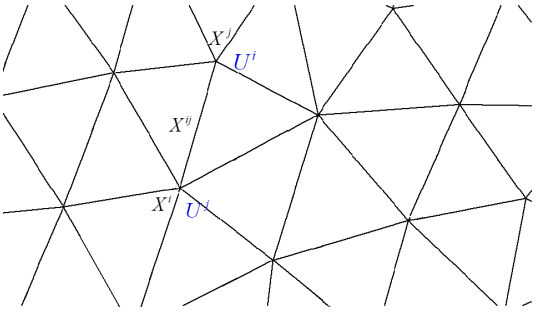


FIGURE 6.1: Length $\mathbf{X}^{ij}$ of the edge joining nodes i and j.

It is now possible to define the length distribution tensor as (see [34] for more details):

$$\mathbb{X}^i = \frac{1}{|\Gamma(i)|} \sum_{j\in\Gamma(i)} \mathbf{X}^{ij} \otimes \mathbf{X}^{ij} \tag{6.2}$$

By the exploitation of a double mean value argument, a hessian based error analysis is conducted, leading to the following definition of error estimator:

$$e_{ij} = \mathbf{g}^{ij} \cdot \mathbf{X}^{ij} \tag{6.3}$$

where $\mathbf{g}^{ij} = \mathbf{g}^{j} - \mathbf{g}^{i}$ and $\mathbf{g}^{i} = \nabla u(\mathbf{X}^{i})$ is the gradient of u evaluated at node $\mathbf{X}^{i}$.

The analysis will not be limited to an analytical function u as discussed in [34], but takes into account the solution of a physical problem. Consequently, the only information readily available on its gradient comes from the finite element approximation, so in a $P1$ setting this means that we do not have access to the point-wise information but only to the element-wise one. For this reason we need to resort to a gradient recovery procedure.

6.2.2 Gradient recovery error estimator

It is shown in [126] that through an optimization analysis it is possible to define a second-order preserving recovered gradient:

$$\mathbf{G}^{i} = (\mathbb{X}^{i})^{-1} \sum_{j \in \Gamma(i)} U^{ij} \mathbf{X}^{ij} \tag{6.4}$$

and the approximated error is then evaluated by substituting $\mathbf{G}$ by $\mathbf{g}$ in (6.3):

$$e_{ij} = \mathbf{G}^{ij} \cdot \mathbf{X}^{ij} \tag{6.5}$$

6.2.3 Metric construction

In this subsection, the tilde will denote the elements associated to the new metric to be computed from the present one. At each node, one may define a new metric by:

$$\widetilde{\mathbb{M}^{i}} = \frac{1}{d} \left(\widetilde{\mathbb{X}^{i}} \right)^{-1}, \tag{6.6}$$

where $\widetilde{\mathbb{X}^{i}}$ is computed as in (6.2) by substituting $\mathbf{X}^{ij}$ with $\widetilde{\mathbf{X}^{ij}} = s_{ij} \mathbf{X}^{ij}$.

A first approach to compute the stretching factors and construct the metric is given in [34]. It consists in minimizing the total error over the mesh. Thus the computation of the stretching factors is done as follows:

$$s_{ij} = \left(\frac{\lambda}{e_{ij}} \right)^{\frac{1}{p+2}} \tag{6.7}$$

and

$$\left(\frac{\sum_i \sum_{j \in \Gamma(i)} e_{ij}^{\frac{p}{p+2}}}{A} \right)^{\frac{p+2}{p}} \tag{6.8}$$

p being a user defined parameter and A the number of edges of the mesh. This anisotropic mesh adpatation method has prooved to be powerful [126]. The authors have investigated the case of the driven cavity up to a Reynolds number of $10,000$ in 2D. They obtain excellent results in terms of accuracy by comparing them to references with a low number of elements. The meshes produced by the method are highly anisotropic and follows the flow vorticities as well as the boundary layers. However this method relies on a user defined parameter, and it is not obvious to define the value of the latter. Moreover it works with the number of edges of the mesh, which is not convenient.

Therefore another method has been developed. As explained in [127], the error changes quadratically with the stretching factors, and from this remark, we proposed a variational method based on a target number of nodes N and a global target equidistributed error e for all lengths. This yields the following definition of the stretching factors:

$$s_{ij} = \left(\frac{e(N)}{e_{ij}} \right)^{\frac{1}{2}} = \left(\frac{\sum_i n^i(1)}{N} \right)^{\frac{2}{d}} e_{ij}^{-1/2}, \tag{6.9}$$

with

$$e(N) = \left(\frac{N}{\sum_i n^i(1)} \right)^{-\frac{4}{d}}$$

the global induced error,

$$n^i(1) = \sqrt{\det \left(\frac{1}{d} \left(\mathbb{X}^i \right)^{-1} \left(\sum_{j \in \Gamma(i)} \left(\frac{1}{e_{ij}} \right)^{-\frac{1}{2}} \mathbf{X}^{ij} \otimes \mathbf{X}^{ij} \right) \right)}$$

the number of new nodes induced by the new edge lengths at node $\mathbf{X}^i$ for a homogeneous error equal to 1 and N the total number of nodes of the mesh.

This method has been improved in [128]. Weights are added in the formulation in order to give a higher importance to the gradient directions. The author shows in different examples that first this new method is able to generate meshes that capture all the scales, and second that it is faster and requires less nodes to converge to the optimal solution. The length distribution tensor becomes:

$$\mathbb{X}^i = \frac{1}{|\Gamma(i)|} \sum_{j \in \Gamma(i)} s_{ij}^2 \omega_{ij} \mathbf{X}^{ij} \otimes \mathbf{X}^{ij} \tag{6.10}$$

with

$$\omega_{ij} = \frac{\|\nabla u \wedge \mathbf{X}_{ij}\|}{\|\nabla u\| \|\mathbf{X}_{ij}\|} \tag{6.11}$$

6.2.4 Control of the L^p norm of the interpolation error

A theoretical validation of the control of the L^p norm of the interpolation error by the second edge based error estimator for a quadratic function is proposed in [128]. The analysis is done in $2D$ over each element K in the mesh. The extension to the $3D$ case is straightforward. Consider a smooth scalar field $u \in \mathcal{C}^2(\Omega)$ and its approximation u_h on the given mesh.

Proposition 6.2.1. Let P be a gauss quadrature point of K. We have:

$$\begin{aligned} \mathbf{X^i P} &= v\mathbf{X^i X^j} + w\mathbf{X^i X^k} \quad \Leftrightarrow \quad P = (1-v-w)X^i + vX^j + wX^k\,, \\ \text{with} \qquad & v = \frac{x_p - x_i}{x_j - x_i} \quad \text{and} \quad w = \frac{y_p - y_i - v(y_j - y_i)}{y_k - y_i} \end{aligned}$$

Consequently, the interpolation error at P can be evaluated in terms of the edge based error estimator:

$$u(P) - \Pi_h u(P) = \frac{v}{2}(v-1)e_{ij} + \frac{w}{2}(w-1)e_{ik} + vwG^{ij}\mathbf{X^{ik}}\,.$$

Proposition 6.2.2. Therefore, the L^p norm of the interpolation error over the domain Ω is given by:

$$\left(\int_\Omega |u - \Pi_h u|^p \, d\Omega\right)^{\frac{1}{p}} = \left(\sum_{K=1}^{n_K} |K| \sum_{g=1}^{n_G} \omega_K^g \left|\frac{v}{2}(v-1)e_{ij}^K + \frac{w}{2}(w-1)e_{ik}^K + vwG_K^{ij}\mathbf{X^{ik}}\right|^p\right)^{\frac{1}{p}}$$

where n_K is the number of elements in the mesh, $|K|$ the volume of the K^{th} element, n_G the number of gauss points for the K^{th} element and ω_K^g the g^{th} quadrature weight for the K^{th} element.

To show the equivalence between the edge based error estimator and the interpolation error in the L_1, L_2 and L_3 norms, we consider the following 2D analytical function

$$u(x,y) = 0,3(x^2 + y^2)$$

The analytical evaluation of the interpolation error is compared to the edge based error estimation computed as described in the previous section. A perfect match between these two and a second order convergence is obtained in the three norms as shown in Figure 6.2.

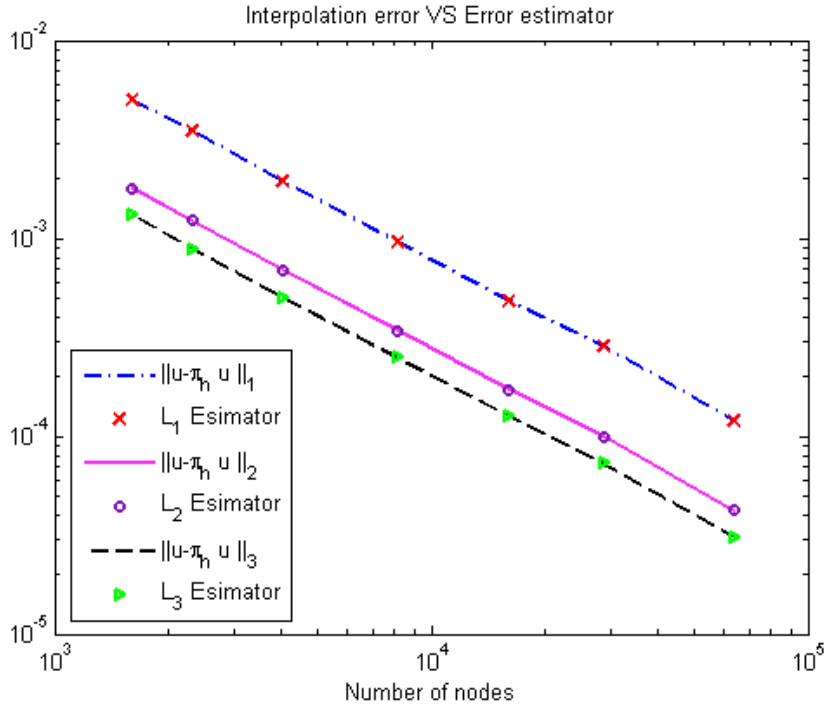


FIGURE 6.2: Rate of convergence of the edge based error estimator and the analytical interpolation error in the L_1, L_2, L_3 norms

6.3 Mesh adaption criteria

In many turbulent flow applications coupled to conjugated heat transfer, the boundary layer and the flow separation need to be modeled accurately. Two strategies are commonly used; 'explicit' and 'implicit' adaptations. In the first one, we design and pre adapt the mesh around the boundaries and in the wake regions by making a priori assumptions about the solution behavior (i.e. Reynolds number, y+, ...). The obtained adapted mesh will be used all over the simulation. The criteria for the mesh adaptation are geometric and do not evolve with the solution. Whereas, the implicit strategy consists in dynamically adapting the mesh and minimizing as much as possible the global equidistributed error. The anisotropic adaptation decisions are in this case entirely driven by the behavior of the *a posteriori* error estimate, taking into account the geometry as well as the evolving solutions.

6.3.1 Comparison with metric intersection on a forced convection case

However, different solutions can be obtained when solving coupled problems; for instance the velocity fields, the pressure distribution and the density evolution in the compressible case. The common way to adapt a mesh to several variables is to compute the metrics corresponding to each of them and then to produce a unique metric by an operation known as the intersection of metrics. In this work, we simplify this operation and we use one metric that accounts for different variables. Therefore, based on the theory proposed in the previous section, we extend the errors e_{ij} and the variable u to become vector fields rather than scalars as proposed in [126]:

$$\mathbf{e}_{ij} = \{e_{ij}^1, e_{ij}^2, \cdots, e_{ij}^n\}$$

n being the number of variables on which the mesh is adapted, and for every node $\mathbf{X}^i$ and direction $\mathbf{X}^{ij}$ we obtain for a given vector field V and a scalar function α:

$$\mathbf{u}(X^i) = \left\{ \frac{V^i}{\|V^i\|_2}, \frac{\|V^i\|_2}{\max\limits_j \|V^j\|_2}, \frac{\alpha}{\max(\alpha)} \right\}.$$

,where α can be the level-set function, the temperature or any other scalar field. Thus, by defining a suitable norm $\|\cdot\|_p$ the corresponding stretching factors become:

$$s_{ij} = \left(\frac{\|\mathbf{e}_{ij}\|_p}{\|\mathbf{e}(N)\|_p} \right)^{-\frac{1}{2}}. \tag{6.12}$$

In what follows we have compared our method of metric construction with metric intersections. We consider the case treated in Chapter 5. Four ingots initially at $20^\circ C$ are heated inside a cavity. Hot air at temperature $1300^\circ C$ is injected through the inlet with a velocity of $10ms^{-1}$ and is evacuated through two outlets placed on the opposite side of the cavity. No turbulence model has been taken into account for this case. The same fields (i.e. temperature, the velocity vector and the level-sets of the four ingots) have been used for the metric intersections, giving three metrics. Two methods have been tested to intersect these three metrics: the maximum eigenvalue [125] and the L_2 norm. The maximum eigenvalue method keeps the minimum mesh size of the three metrics at a given node whereas the L_2 norm method computes the L_2 norm of the three metrics. Figures 6.4 to 6.8 compare the edge-based error estimator method with the two methods used for the metrics intersection at different times of the simulation. The temperature and the velocity fields as well as the mesh are presented for each method. Figures 6.5 and 6.7 show that the meshes produced by the edge-based estimator and the metric intersection are radically different. While the metric intersection focuses on the temperature, the edge-based estimator follows also the velocity field and produces a very fine anisotropic mesh on the boundary layers. Figure 6.9 shows a zoom up to

$\times 100,000$ on the uppper part of the cavity on the boundary layer zone. This confirms the ability of the developed algorithm to generate highly directional stretched elements. The mesh produced by the edge-based estimator captures better all the vortices as well, as reflected in Figure 6.8. We conclude this study with computational time data. Table 6.2 presents the computational time of the three different metric construction method at time $t = 10s$ and $t = 200s$. The metric intersection method using the L_2 norm is clearly faster, whereas the one using the maximum eigenvalue is the less efficient. It is worth mentioning that the same number of nodes has been used for each computation (approximately $15,000$). Each computation has been run on 10 cores (AMD 64 $2.5GHz$). Given these observations we will use the edge-based error estimator to construct the metric in the rest of the computations. Since the computational time is slightly higher than the one with the metric intersection method using the L_2 norm, the produced mesh has a better quality and captures better the physics of the problem.

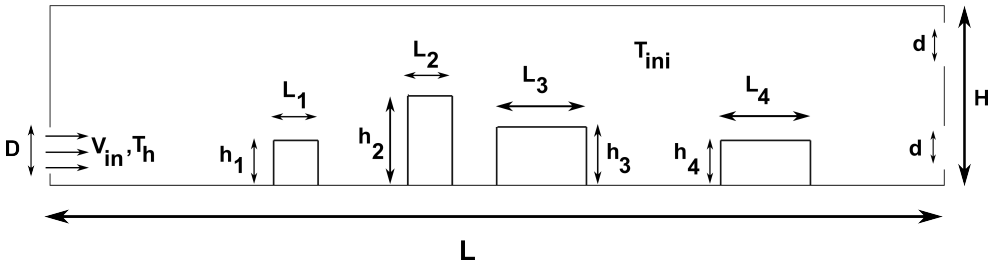


FIGURE 6.3: Case for the comparison of the edge-based metric and the intersection of metrics

	cavity	Ingot 1	Ingot 2	Ingot 3	Ingot 4
Dimensions (Lxh) (m)	20×4	1×1.5	1×2.5	2×1.8	2×1.5
ρ $(kg.m^{-3})$	1	$1,000$	$2,000$	$8,000$	$6,000$
Cp $(J.kg^{-1}.K^{-1})$	$1,000$	500	500	500	500
ν $(kg.m^{-1}.s{-}1)$	2.10^{-5}	10^5	10^5	10^5	10^5
k $(W.m^{-1}.K^{-1})$	0.0262	26	26	26	26

TABLE 6.1: Physical properties of the four ingots and the cavity

	$t = 10s$	$t = 200s$
Edge-based estimator	13	733
Metric intersection (max eigenvalue)	90	$1,584$
Metric intersection (L_2 norm)	7	225

TABLE 6.2: Computational time in minutes of the three different metric construction methods

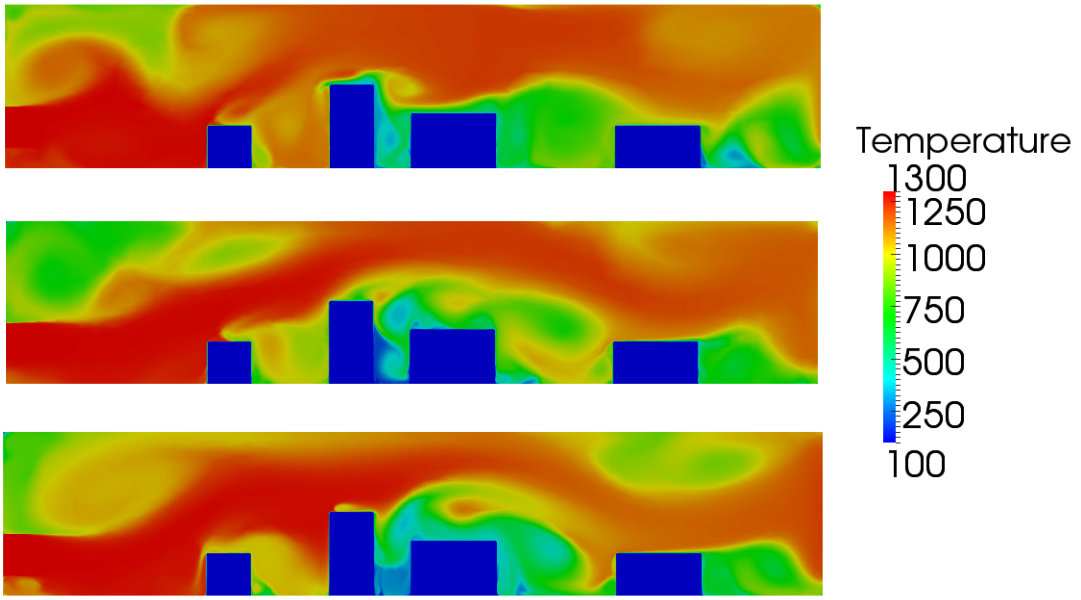


FIGURE 6.4: Temperature field in $^{\circ}C$ with the different adaptation methods at time $t = 10s$: edge-based estimator (top), metric intersection with maximum eigenvalue (middle) and metric intersection with L_2 norm

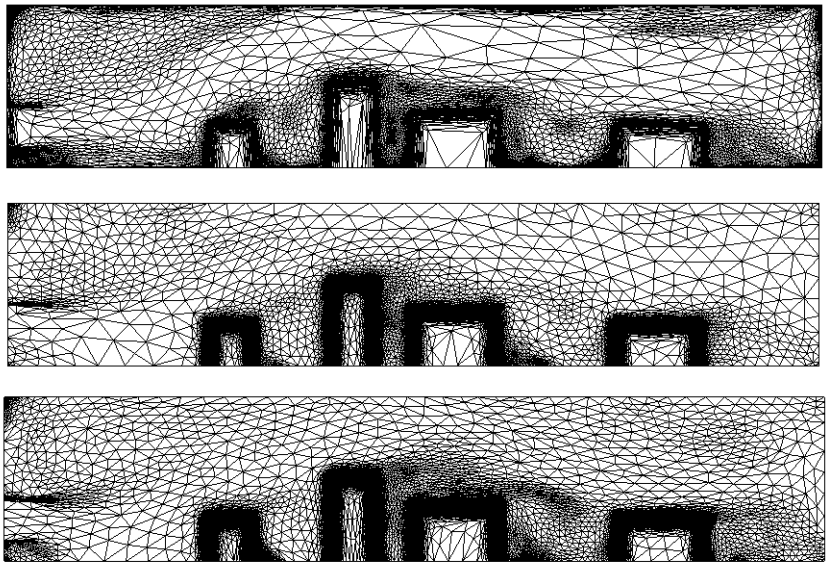


FIGURE 6.5: Mesh of the different adaptation methods at time $t = 10s$: edge-based estimator (top), metric intersection with maximum eigenvalue (middle) and metric intersection with L_2 norm

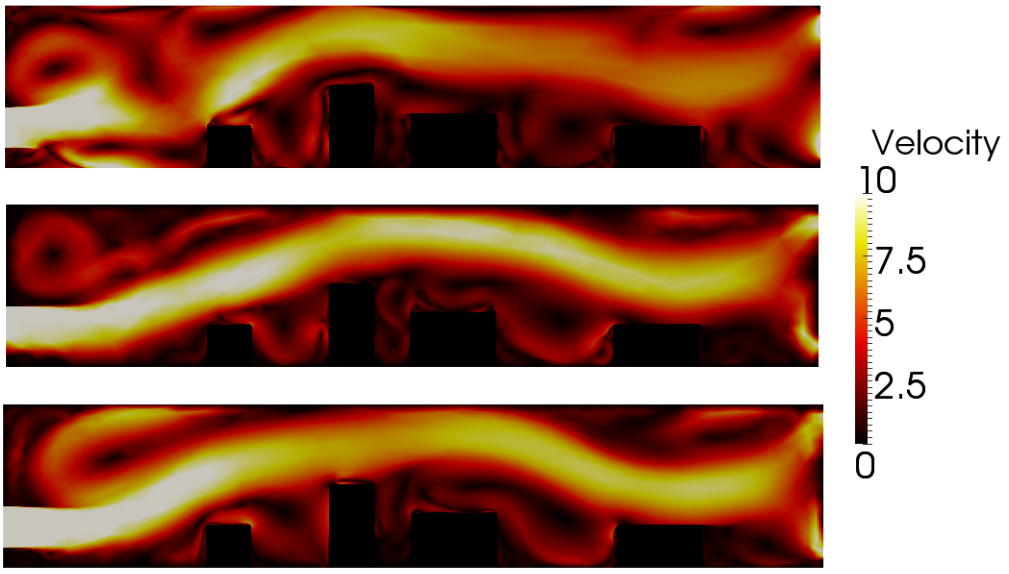


FIGURE 6.6: Velocity field in ms^{-1} with the different adaptation methods at time $t = 10s$: edge-based estimator (top), metric intersection with maximum eigenvalue (middle) and metric intersection with L_2 norm

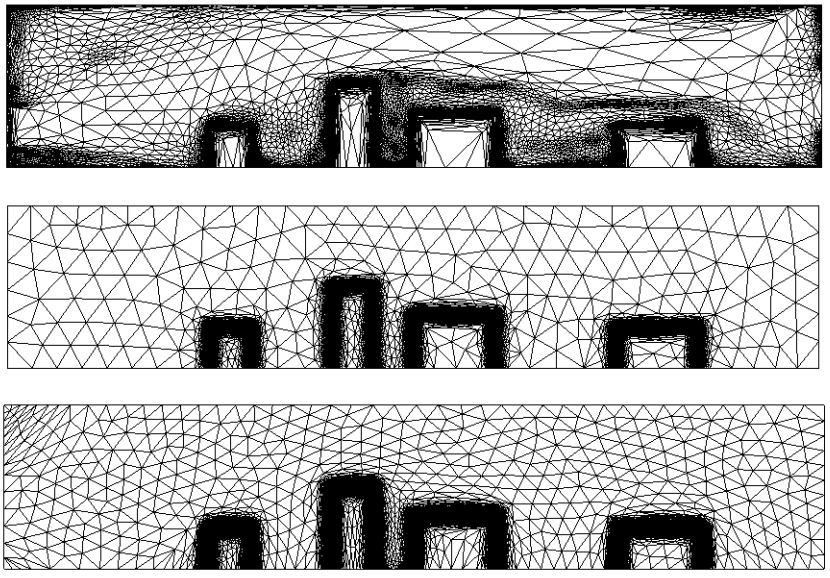


FIGURE 6.7: Mesh of the different adaptation methods at time $t = 200s$: edge-based estimator (top), metric intersection with maximum eigenvalue (middle) and metric intersection with L_2 norm

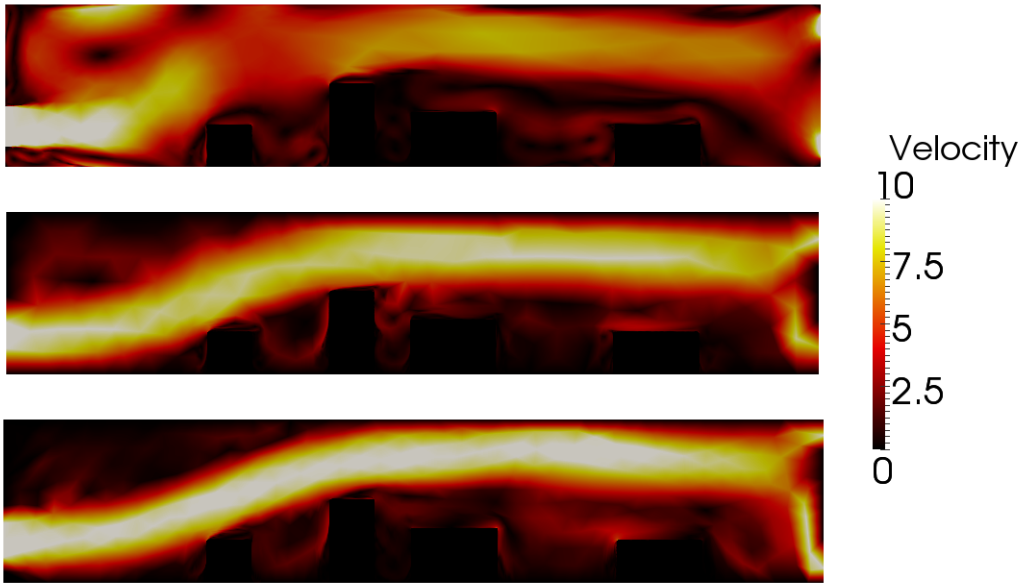


FIGURE 6.8: Velocity field in ms^{-1} with the different adaptation methods at time $t = 200s$: edge-based estimator (top), metric intersection with maximum eigenvalue (middle) and metric intersection with L_2 norm

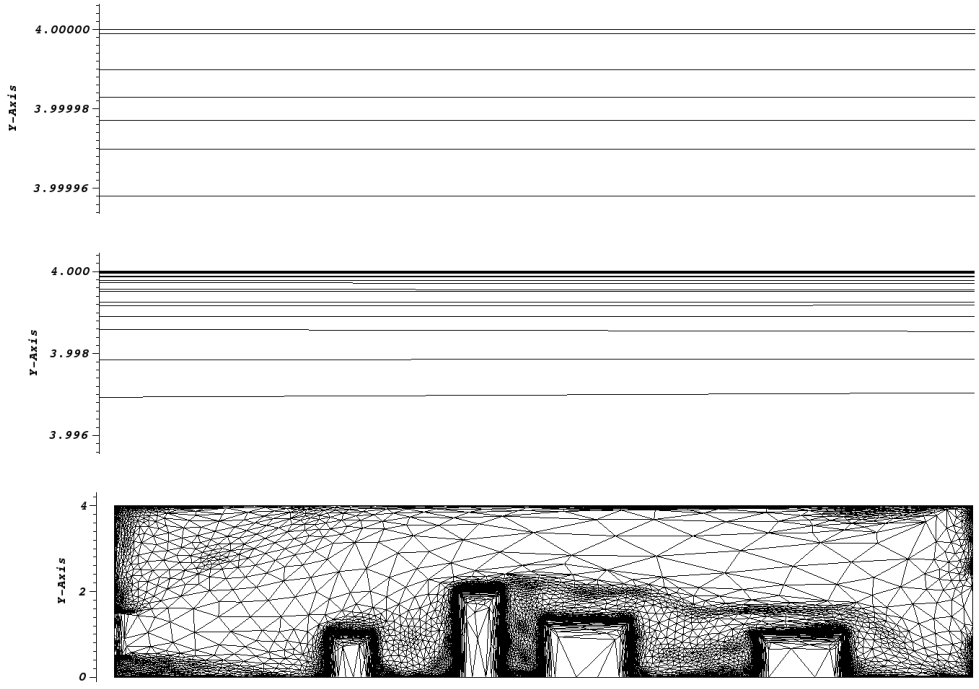


FIGURE 6.9: Zoom on the mesh of the edge-based estimator method at time $t = 200s$: $\times 1$ (top), $\times 1,000$ (middle) and $\times 100,000$

6.3.2 Multi-criteria applied to a natural convection case

In this section, we discuss the performance of the multi-criteria edge-based error adaptive anisotropic mesh method for heat transfer and fluid flows. We analyze a very well known benchmark on the natural convection in 2D and 3D. The obtained results are compared to a set of references.

6.3.2.1 Natural convection benchmark (2D)

We start by establishing a reference solution for Rayleigh numbers ranging from 10^4 to 10^8. This reference solution is computed using a fixed isotropic mesh of 10^6 elements. To compare and assess the accuracy of the reference solution solved using the stabilized finite element methods, we compare the obtained Nusselt number with results found in the litterature. Next, we apply the mesh adaptation with the mutli-criteria approach using the velocity components, the temperature or their combinations, highlighting the capability of the method for coupled problems.

We consider a unity square cavity (Figure 6.10) with a thermal gradient applied to the vertical walls whereas the horizontal walls are insulated. No slip boundary condition is imposed on all the walls. The left wall is maintained at a temperature T_h and the right wall at a temperature T_c, with $T_h > T_c$. Five computations have been conducted for

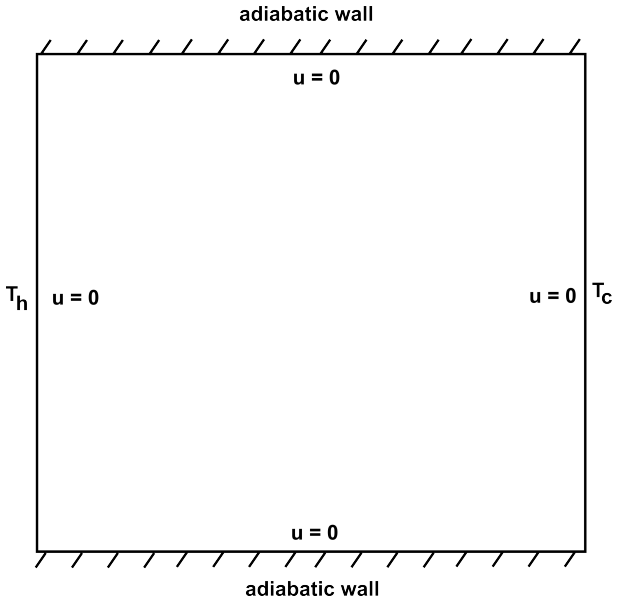


FIGURE 6.10: Natural convection boundary conditions in the square cavity

Rayleigh numbers varying from 10^4 to 10^8. Note that here we solved the Navier-Stokes equations under the classical Boussinesq approximation. In order to verify the accuracy

of the reference solution, we compare the results of each computation to the ones found in the litterature in terms of Nusselt number. The latter is given by:

$$Nu = \int_0^L \frac{\partial T}{\partial x} dy, \ at \ x = 0$$

Ra	10^4	10^5	10^6	10^7	10^8
[129]	2.238	4.509	8.817	-	-
[130]	2.201	4.430	8.754	-	32.045
[131]	-	-	8.822	-	30.200
[132]	-	-	8.860	16.241	-
[133]	2.247	4.523	8.805	16.790	30.506
Present	2.274	4.599	8.827	16.829	31.840

TABLE 6.3: Comparison of the Nusselt number in the present study with the litterature

Table 6.3 shows that the obtained results are in very good agreement with the litterature. The small difference may be due to the used discretization methods (Lattice Boltzmann method, finite difference,...) or the used density variation model (compressible flows, Boussinesq approximation,...). Next, we investigate different fields for adapting the mesh. The objective is to find the best combination that increases the quality of the solution and its accuracy. We propose as adaptation criteria the following expression: $(\omega_1 \frac{T}{max(T)}, \omega_2 \frac{v}{max(\|v\|_2)}, \omega_3 \frac{\|v\|_2}{max(\|v\|_2)})$, with $\frac{T}{max(T)}$ the normalized temperature, $\frac{v}{max(\|v\|_2)}$ the normalized velocity components and $\frac{\|v\|_2}{max(\|v\|_2)}$ its normalized magnitude. Using the same number of elements, we compare to the reference solution established previously for different Rayleigh numbers three different configurations: $(\omega_1, \omega_2, \omega_3) = (1, 0, 0)$, $(\omega_1, \omega_2, \omega_3) = (0, 1, 1)$ and $(\omega_1, \omega_2, \omega_3) = (1, 1, 1)$. Table 6.4 summarizes the number of elements corresponding to the different cases.

Ra	10^4	10^5	10^6
Reference	$1,000,000$	$1,000,000$	$1,000,000$
Mesh adaptation	$9,000$	$17,000$	$24,000$

TABLE 6.4: Number of used elements for different Rayleigh numbers

The results obtained using multiple-criteria and anisotropic mesh adaptation are compared to the reference solution for different Rayleigh numbers. In Figure 6.11, we present the first three plots for $Ra = 10^4$ along the x-axis and y-axis for the velocity components and the temperature solutions respectively. As expected, all the solutions agreed very well despite the significant differences noticed in the obtained meshes as shows Figure 6.12. As the Rayleigh number increases (10^5 and 10^6), the convective terms becomes more dominant leading to some variations in the solutions. Figures 6.13 and 6.15 confirm that choosing the normalized temperature $\frac{T}{T_{max}}$ as a criterion for the mesh adaptation

lead always to more accurate results. This is noticed on all the plots, in particular for the y-component of the velocity field. Again, the difference between the obtained meshes becomes more clear and interesting for higher Rayleigh numbers. We can clearly see in Figures 6.14 and 6.16 that the adaptation along the temperature field results a much denser and finer meshes concentrated near high temperature gradients and close to the vertical walls. Whereas, when using the velocity as the mesh adaptation criteria, extremely stretched elements are noticed and all the boundary layers are sharply captured and automatically identified. Note the concentration of the mesh is not only along all the boundary layers but also at some detachment regions close to the center. This reflects well the anisotropy of the solution caused by the discontinuity of the boundary conditions and the nature of the flow. The elements far from the central bulk of the cavity are mostly isotropic and increase in size as the Rayleigh number increases. This explains how, for a controlled number of nodes, the mesh is naturally and automatically coarsened in that region with the goal of reducing the mesh size around the high gradients in the solutions. It is mainly due to the proposed mutli-criteria strategy of using one unit vector combining multiple components.

The proposed approach shows that it is capable of capturing all the physics at all scales. Indeed, small vortices, detachments and thin boundary layers are usually critical to be captured accurately and efficiently by isotropic meshes. We have thereby showed that the anisotropic mesh adaptation method works well in capturing the characteristics of the solution and is able to give accurate results when solving the coupled heat transfer and fluid flows equations. We have also established that adapting the mesh on the normalized temperature $\frac{T}{T_{max}}$ provides better results for this benchmark and with the prescribed num of elements, especially for high Rayleigh number. In fact when adapting the mesh on the velocity, the highest gradients are localized neer the boundary layers, thus the estimated error is higher in this region. Therefore the mesh is extremely densified there and coarsened in the rest of the cavity, leading to a small drift in the result curves. In order to get a finer mesh out of the boundary layers and thus more accurate results, the number of elements has to be increased. Therefore adapting the mesh on the temperature field with the prescribed number of elements provides better results out of the boundary layers whereas adapting the mesh on the velocity field gives more accurate results near the boundary layers.
Finally, in order to assess the capability of the proposed method to simulate higher Rayleigh number flows with anisotropic meshes, we repeated the simulation for Rayleigh number of 10^8. For more efficient comparisons, we have added to the existing plots along the centerlines some additional plots close to the boundary layers: $x = 0.05$, $y = 0.05$, $x = 0.95$ and $y = 0.95$. The number of elements was fixed to $40,000$ elements. Based on the previous results, we considered only the normalized temperature $\frac{T}{T_{max}}$ as the adaptation criterion. Figures 6.17 to 6.19 compare all the obtained results at different cuts and highlight the accuracy of the proposed anisotropic mesh adaptation. The

velocity peaks and the temperature profiles are well captured. The accuracy is conserved near the boundary layers, especially along the lines $x = 0.05$ and $x = 0.95$. Recall that in these regions, the obtained mesh is extremely fine due the steep temperature gradient (see Figure 6.20).

In order to push the investigation further, we have also computed the benchmark solution for a Rayleigh number of 10^9. The computation has been run using $40,000$ elements. The mesh is adapated only on the temperature field. With such a Rayleigh number, the flow no more converges to a steady state and keeps changing in time. Figure 6.21 shows the temperature evolution at different times. We can notice that the temperature is convected from one side of the domain to the other, and progressively diffuses to the center of the cavity. Figure 6.22 presents the associated adapted meshes. The mesh follows dynamically the temperature field in order to keep an accurate solution. Finally Figure 6.23 shows the streamlines of the flow. The flow is turbulent and several vortices are created. We can notice how the flow is chaotic and that the vortices are localized at different positions in time.

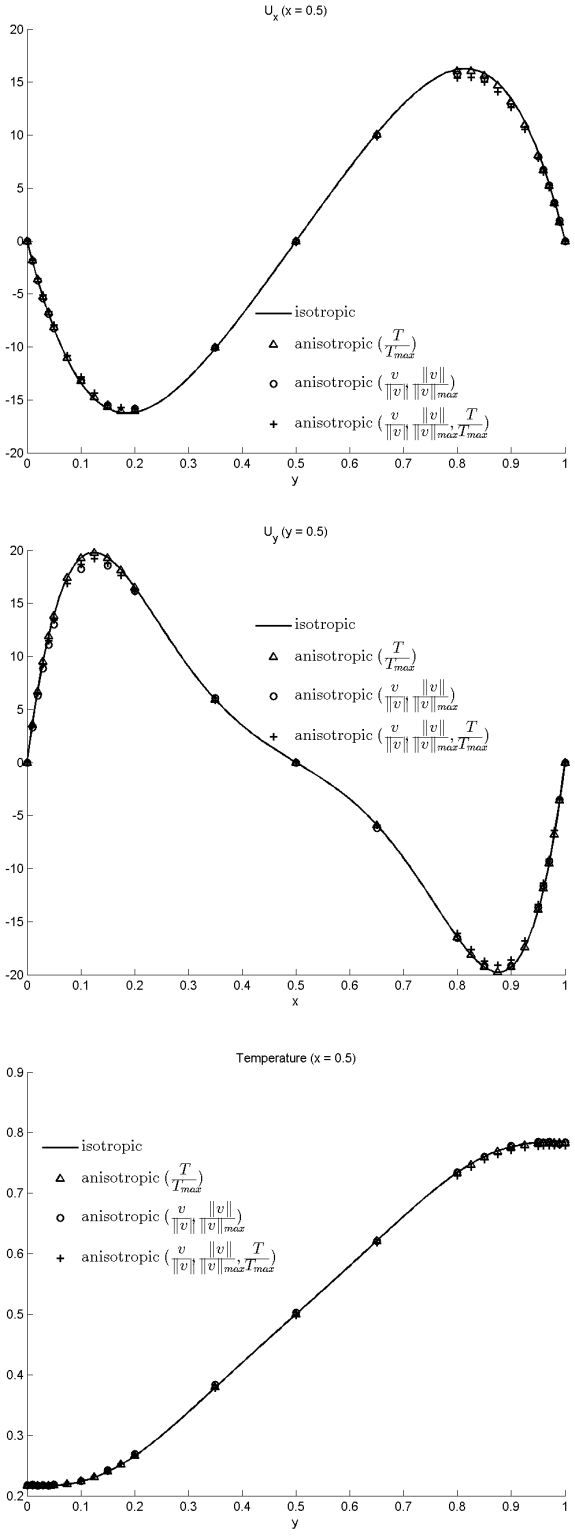


FIGURE 6.11: Comparison of the mesh adaptation field on the x velocity component (top left), the y velocity component (top right) and the temperature (bottom), $Ra = 10^4$

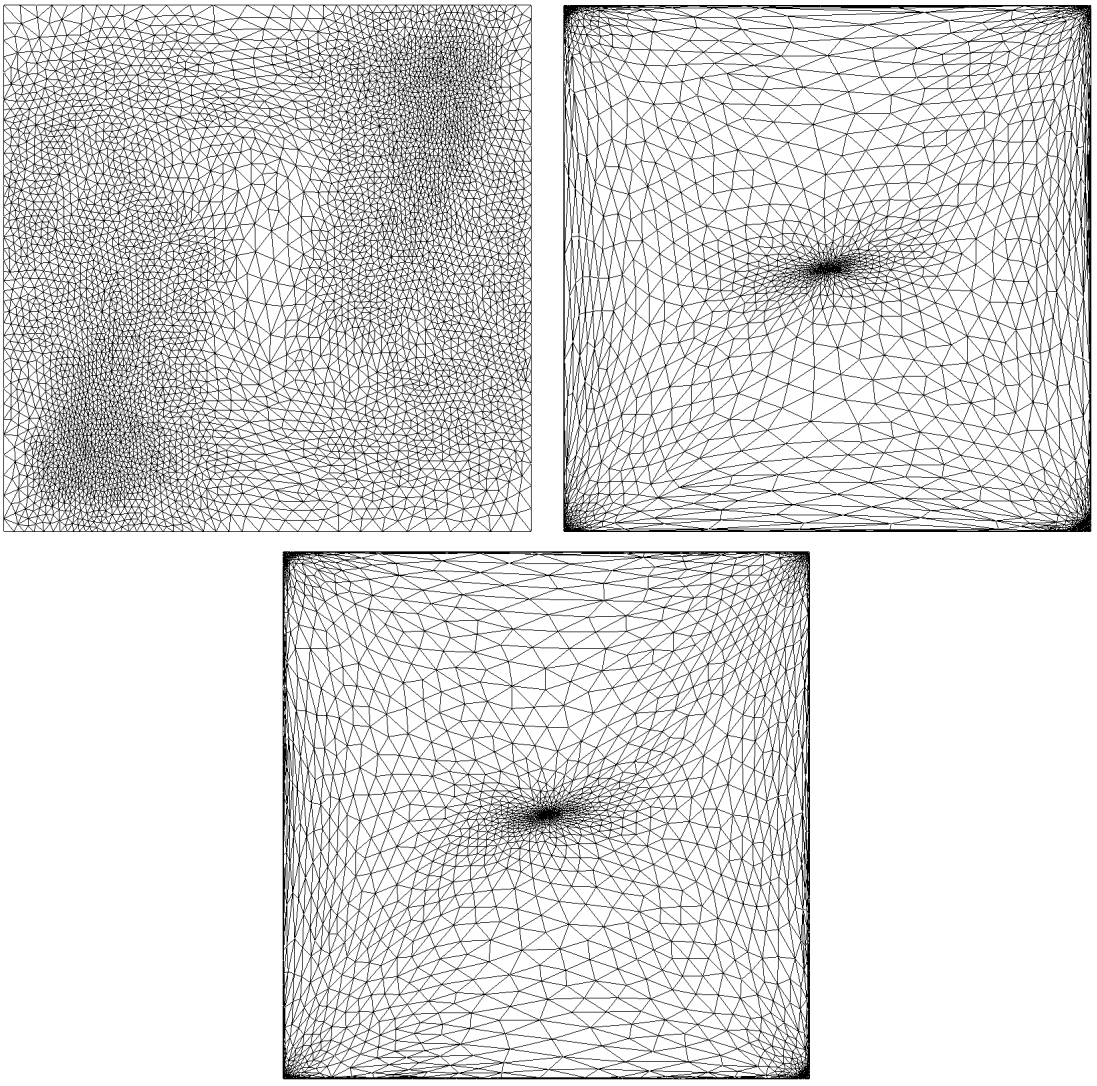


FIGURE 6.12: Comparison of the adapted mesh in function of the mesh adaptation field $\frac{T}{T_{max}}$ (left), $\frac{v}{\|v\|}$, $\frac{\|v\|}{\|v\|_{max}}$ (middle), $\frac{T}{T_{max}}$, $\frac{v}{\|v\|}$, $\frac{\|v\|}{\|v\|_{max}}$ (right), $Ra = 10^4$

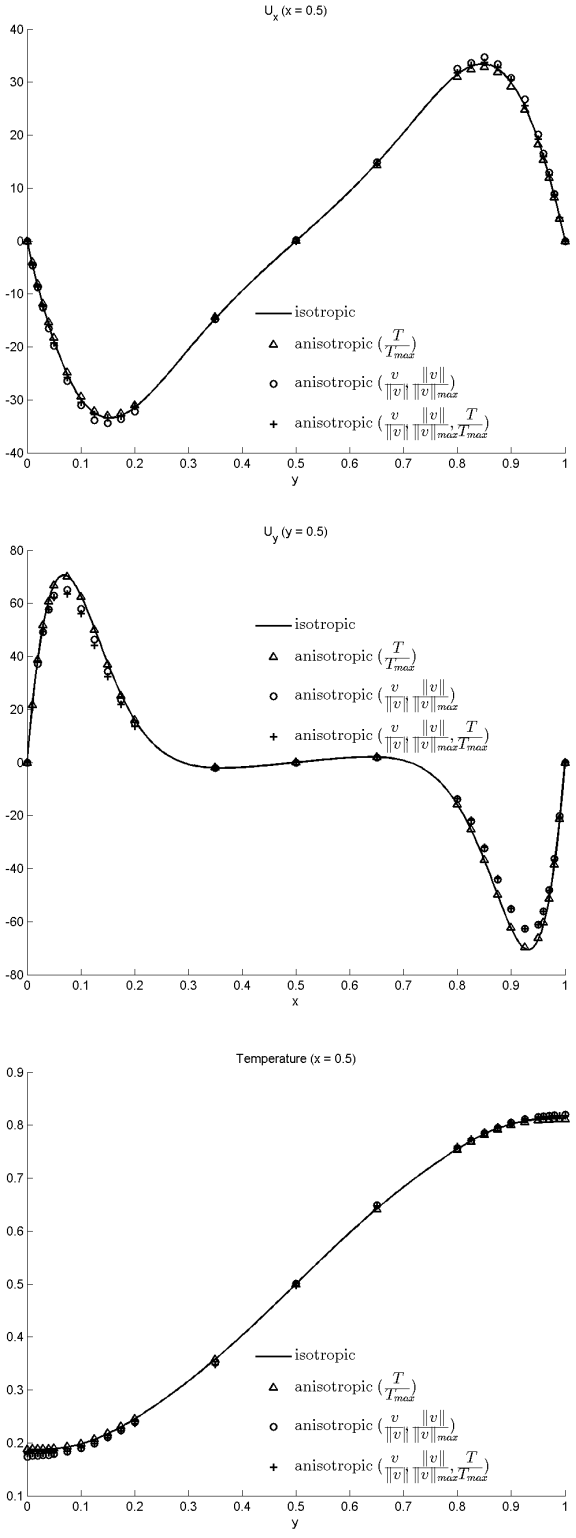


FIGURE 6.13: Comparison of the mesh adaptation field on the x velocity component (top left), the y velocity component (top right) and the temperature (bottom), $Ra = 10^5$

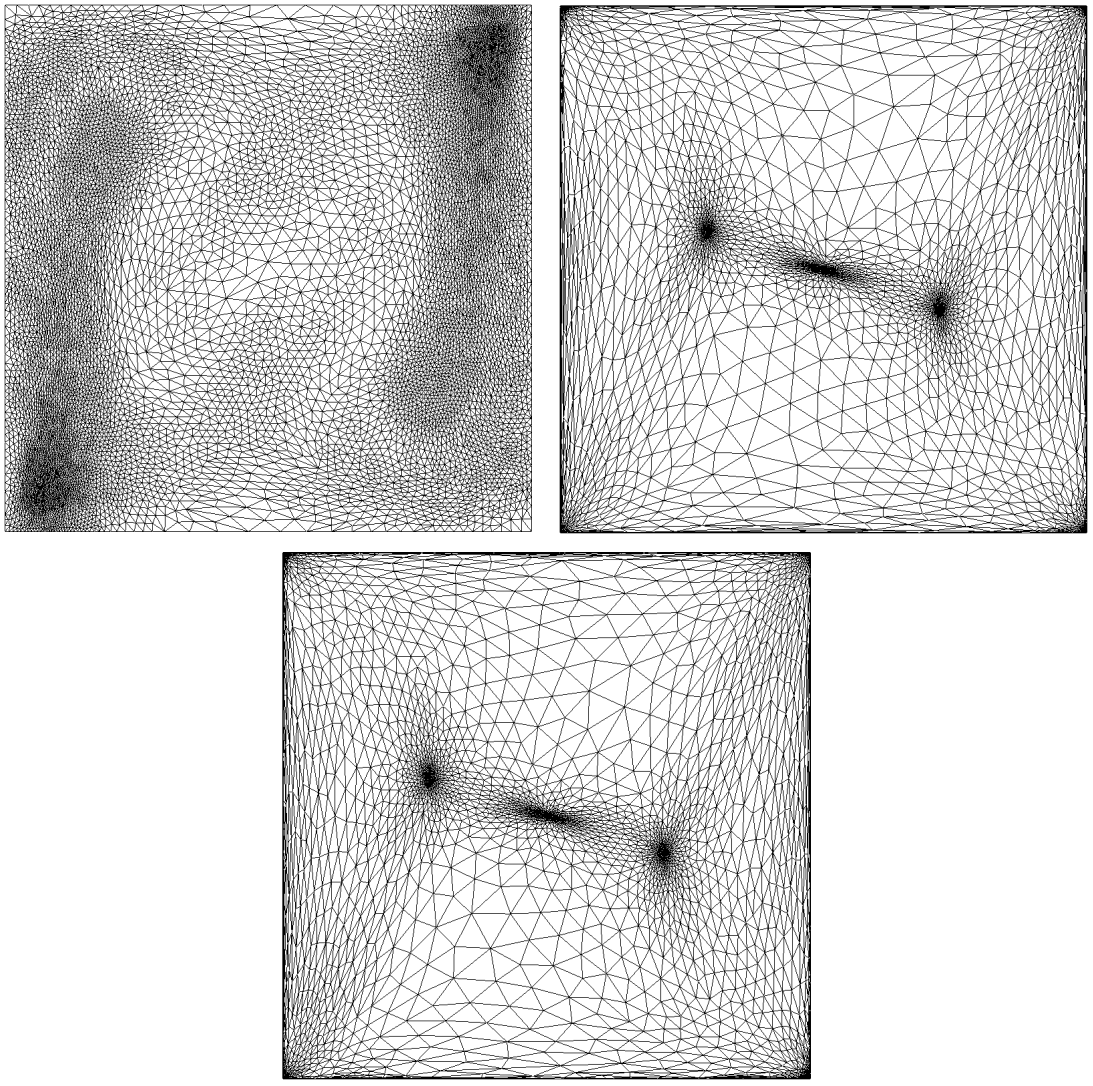


FIGURE 6.14: Comparison of the adapted mesh in function of the mesh adaptation field $\frac{T}{T_{max}}$ (left), $\frac{v}{\|v\|}$, $\frac{\|v\|}{\|v\|_{max}}$ (middle), $\frac{T}{T_{max}}$, $\frac{v}{\|v\|}$, $\frac{\|v\|}{\|v\|_{max}}$ (right), $Ra = 10^5$

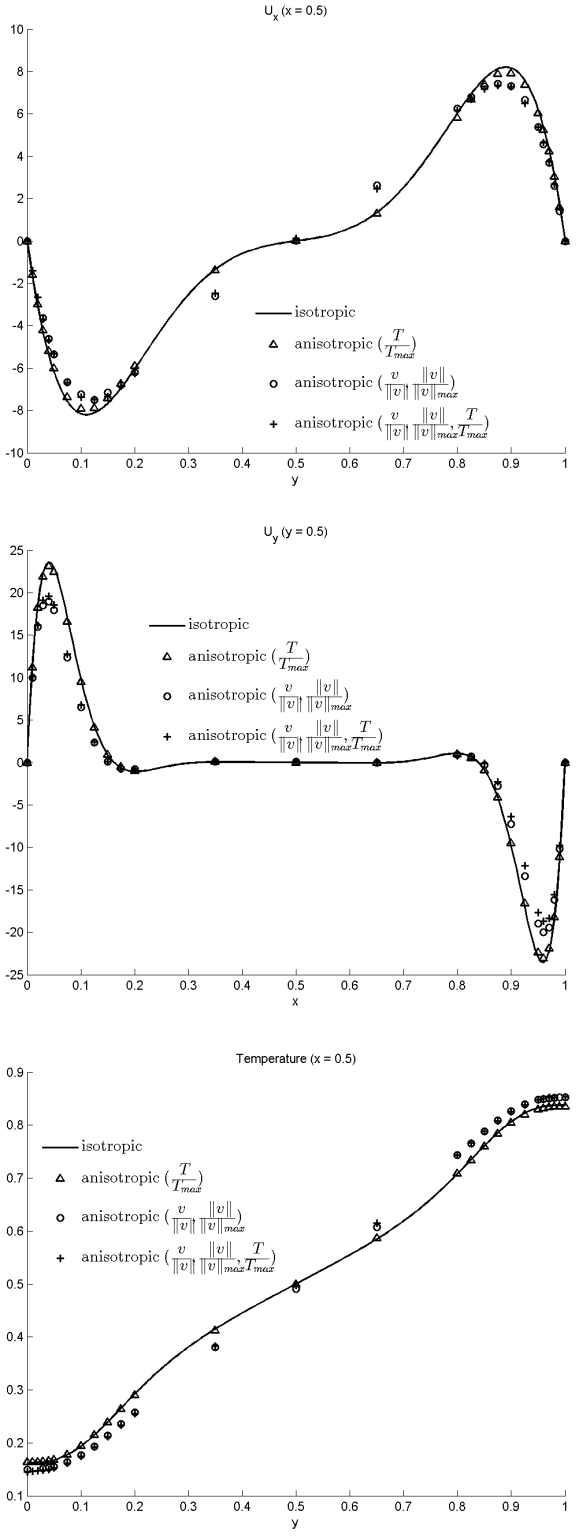


FIGURE 6.15: Comparison of the mesh adaptation field on the x velocity component (top left), the y velocity component (top right) and the temperature (bottom), $Ra = 10^6$

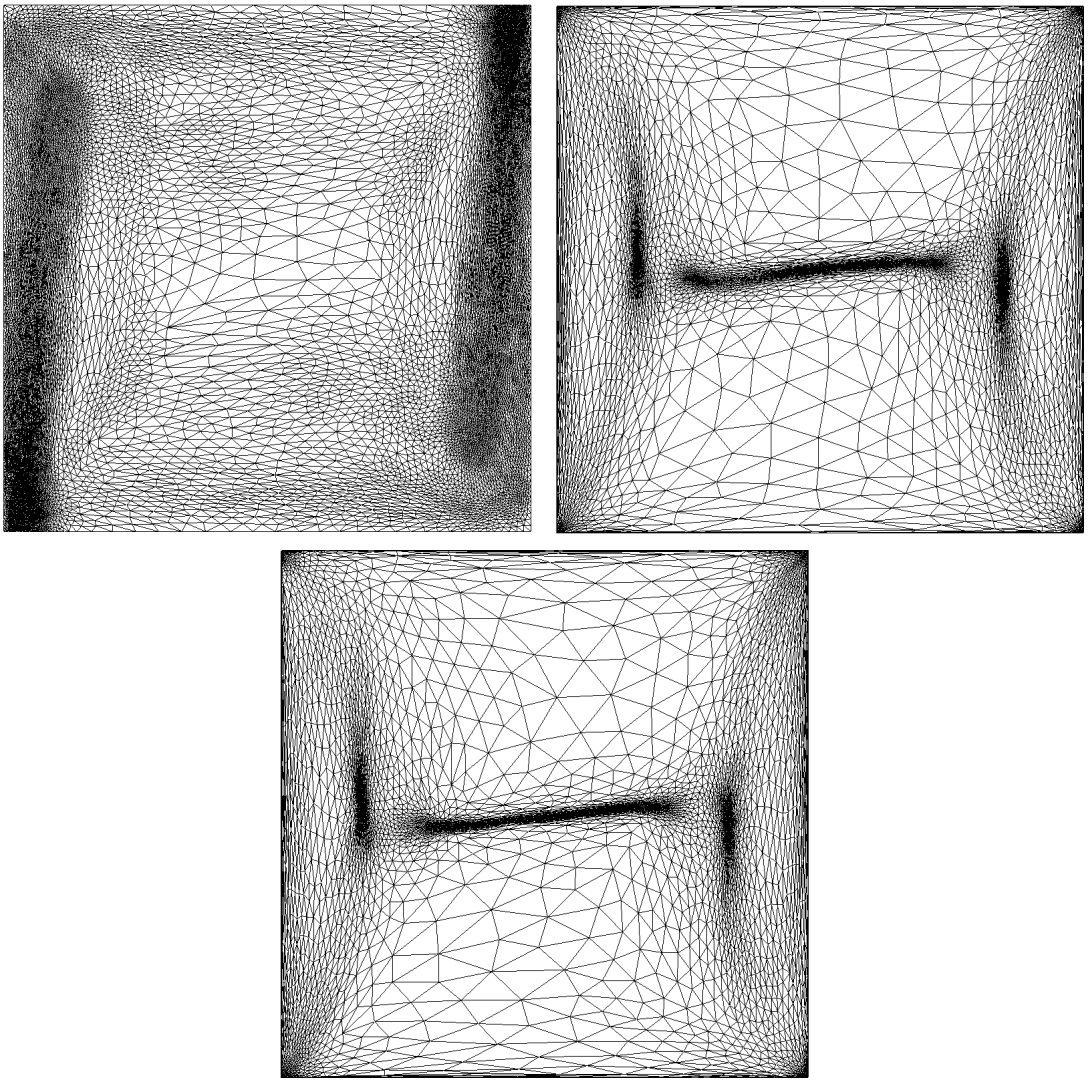


FIGURE 6.16: Comparison of the adapted mesh in function of the mesh adaptation field $\frac{T}{T_{max}}$ (left), $\frac{v}{\|v\|}$, $\frac{\|v\|}{\|v\|_{max}}$ (middle), $\frac{T}{T_{max}}$, $\frac{v}{\|v\|}$, $\frac{\|v\|}{\|v\|_{max}}$ (right), $Ra = 10^6$

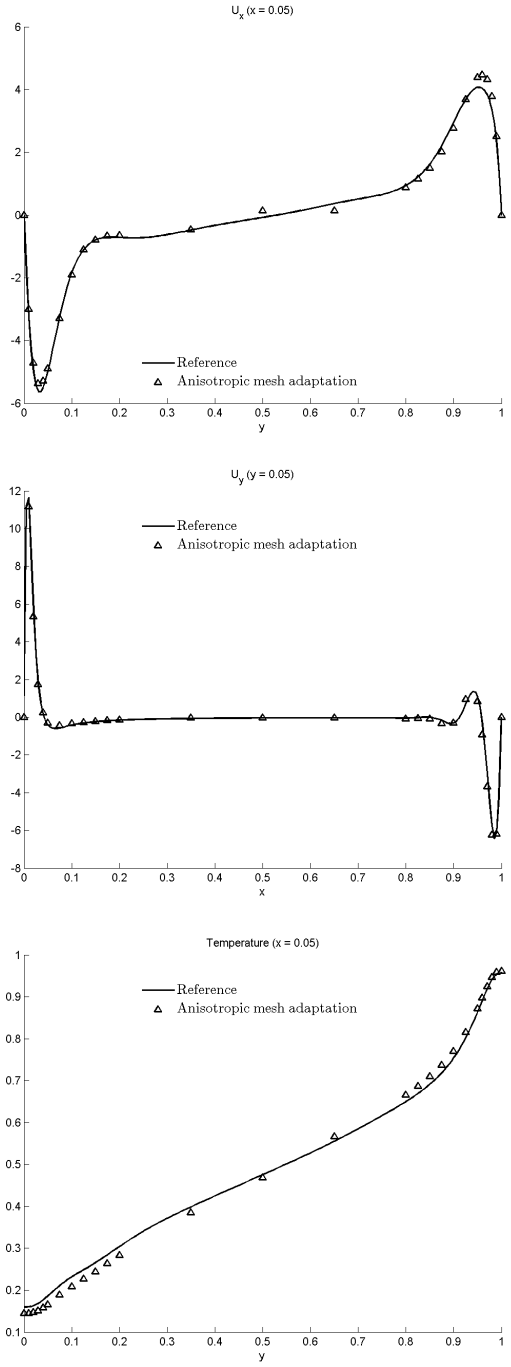


FIGURE 6.17: Comparison of the anisotropic mesh adaptation method to the reference solution at x=0.05 and y=0.05 ($\frac{T}{T_{max}}$ was used for the mesh adaptation), $Ra = 10^8$

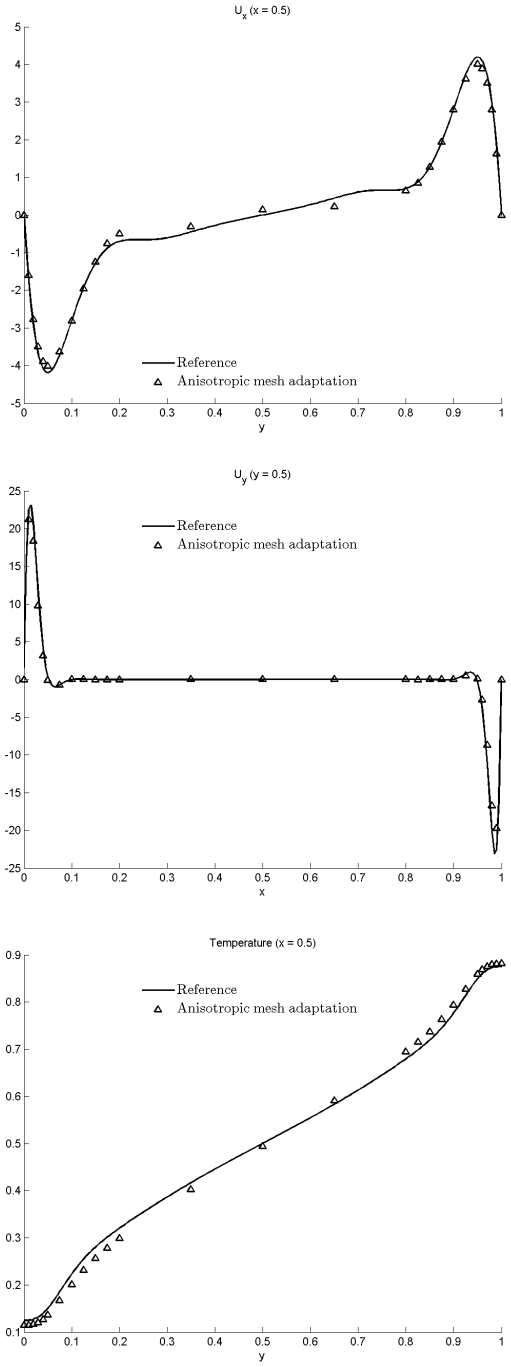


FIGURE 6.18: Comparison of the anisotropic mesh adaptation method to the reference solution at x=0.5 and y=0.5 ($\frac{T}{T_{max}}$ was used for the mesh adaptation), $Ra = 10^8$

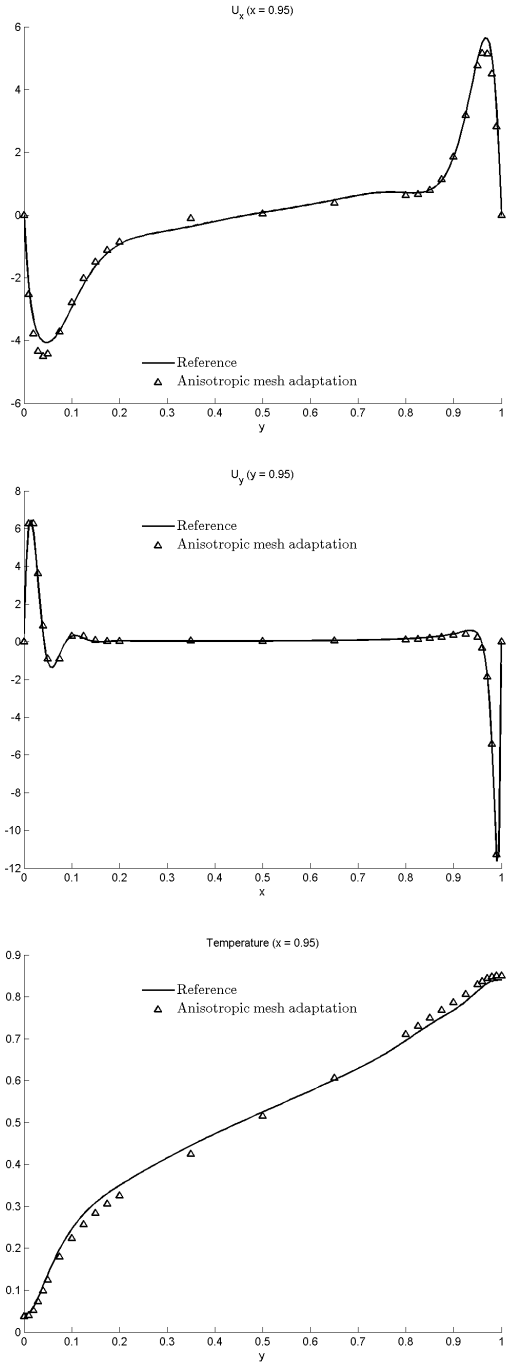


FIGURE 6.19: Comparison of the anisotropic mesh adaptation method to the reference solution at x=0.95 and y=0.95 ($\frac{T}{T_{max}}$ was used for the mesh adaptation), $Ra = 10^8$

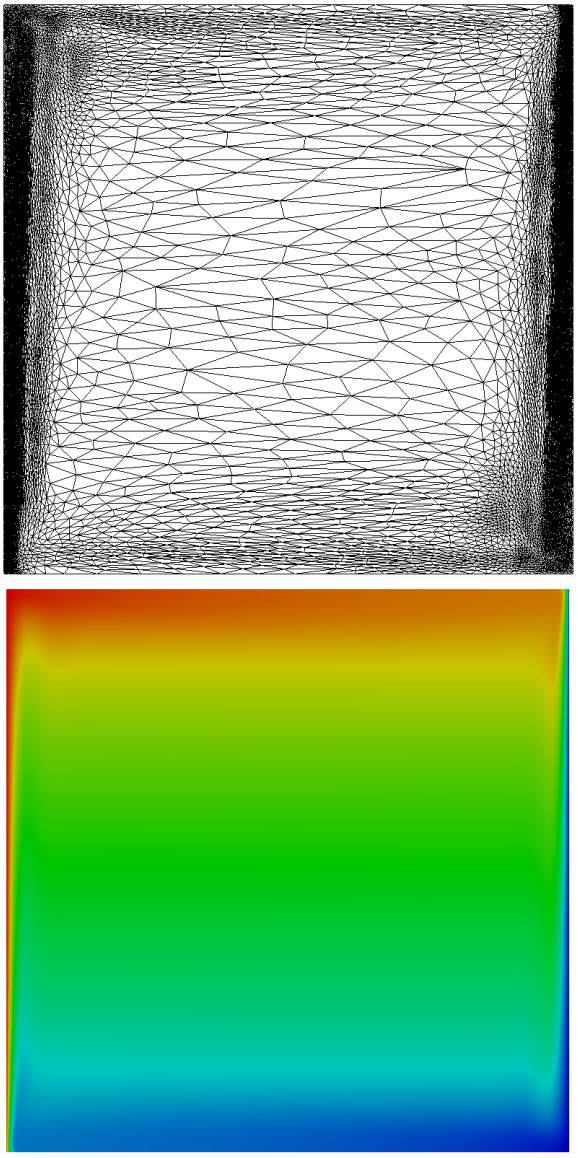


FIGURE 6.20: Adapted mesh on field $\frac{T}{T_{max}}$ (top) and isotherms (bottom), $Ra = 10^8$

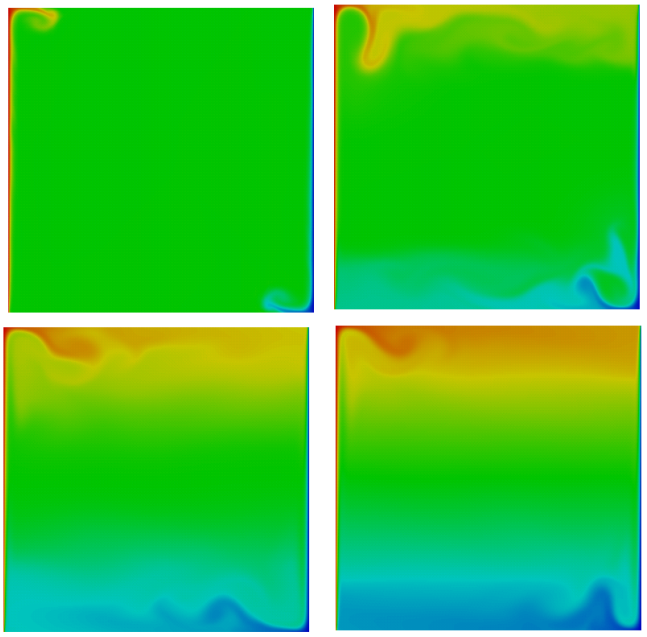


FIGURE 6.21: Temperature distribution through the cavity at different times, $Ra = 10^9$

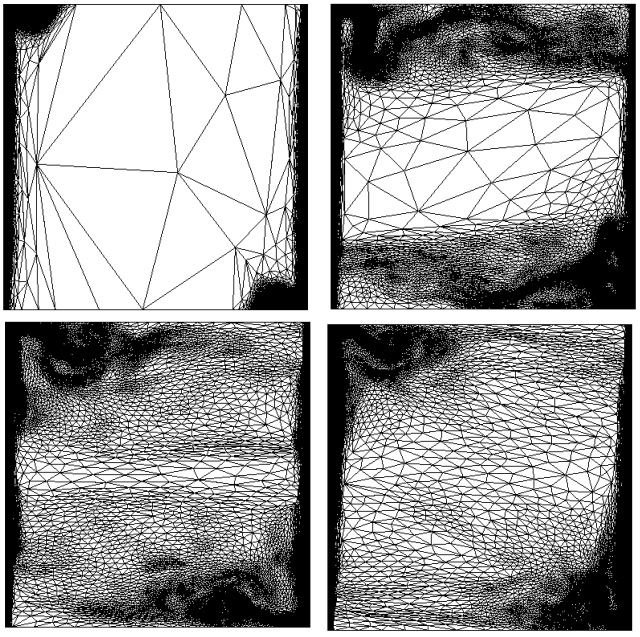


FIGURE 6.22: Adapted meshes on the temperature field at different times, $Ra = 10^9$

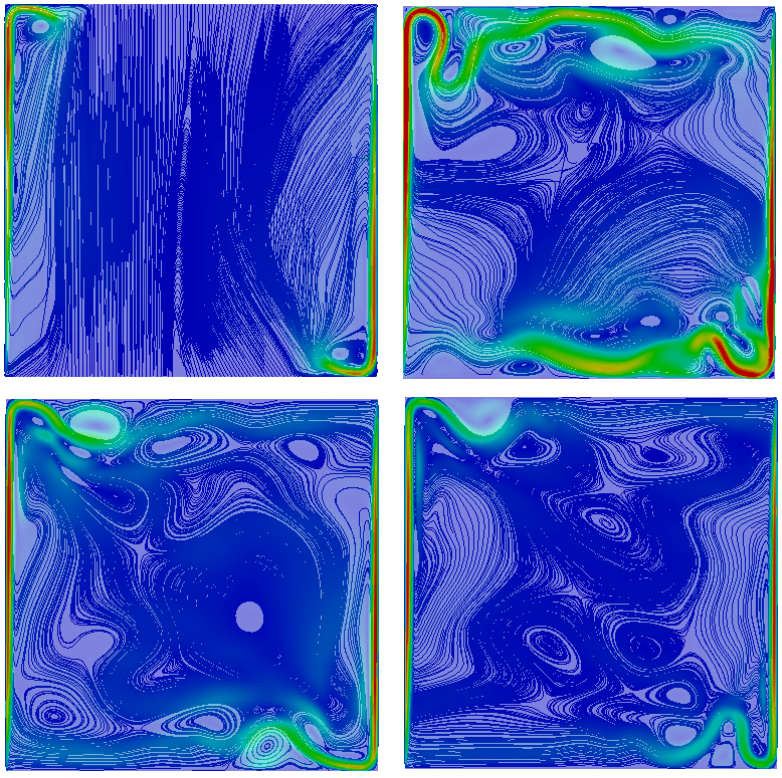


FIGURE 6.23: Flow streamlines at different times, $Ra = 10^9$

6.3.2.2 Natural convection benchmark (3D)

We continue to numerically solve this classical benchmark in 3D. Same as in the 2D case, no slip boundary conditions are imposed at the walls. The horizontal walls are insulated and the vertical walls are maintained at constant but different temperature. Indeed, the left wall is forced to stay hot while the right wall is forced to stay cold. All numerical experiments are done with a fixed number of elements (˜100,000 elements) for Rayleigh numbers ranging from 10^4 to 10^8. The mesh is adapted on the normalized temperature field ($\frac{T}{T_{max}}$). Figure 6.24 shows the velocity streamlines and the characteristics of the flow becoming more complicated as the Rayleigh numbers increases. It also highlights the refinement of the elements near the walls and close to the corners, allowing a better capture of the temperature gradients. This reflects the precision of the solution which is in accordance to the high resolution of the mesh at the boundary layers. The anisotropic adaptive procedure modifies the mesh in a way that the local mesh resolutions become adequate in all directions. Recall again that these figures reflect for the given fixed number of elements the mesh that maximizes the accuracy of the numerical solution. More analysis taking into account an increased number of nodes and comparisons with an extra resolved reference solution will be the subject of further investigations. The presented test cases are considered here in the objective of demonstrating the capability of the method to simulate 2D and 3D high Rayleigh number flows on anisotropic meshes.

6.4 Conclusion

In this chapter, we introduced the anisotropic mesh adaptation method. We presented a method which is based on the edges of the mesh. It consists in first computing an error along each edge by recovering the gradient with the nodal solution and the length distribution tensor. Then from the error stretching factors are computed and a new metric is built up. The method is original as it naturally deals with multi-components for the mesh adaptation criteria. An error is computed for each criterion of the mesh adaptation and a L_p norm is calculated on each edge. This natural treatment of the multi-criteria better takes into account each field of adaptation (temperature, velocity, etc...) and generates a mesh that has a better quality and better captures the physics of the problem compared to metric intersection methods. Different criteria haven been used on a 2D natural convection benchmark in order to analyze the impact of the produced mesh as well as the precision of the solution. The method has shown its capability to recover an accurate solution at high Rayleigh numbers (up to 10^8). This also demonstrates the performance of the solvers to compute a solution of high precision with extremely stretched elements.

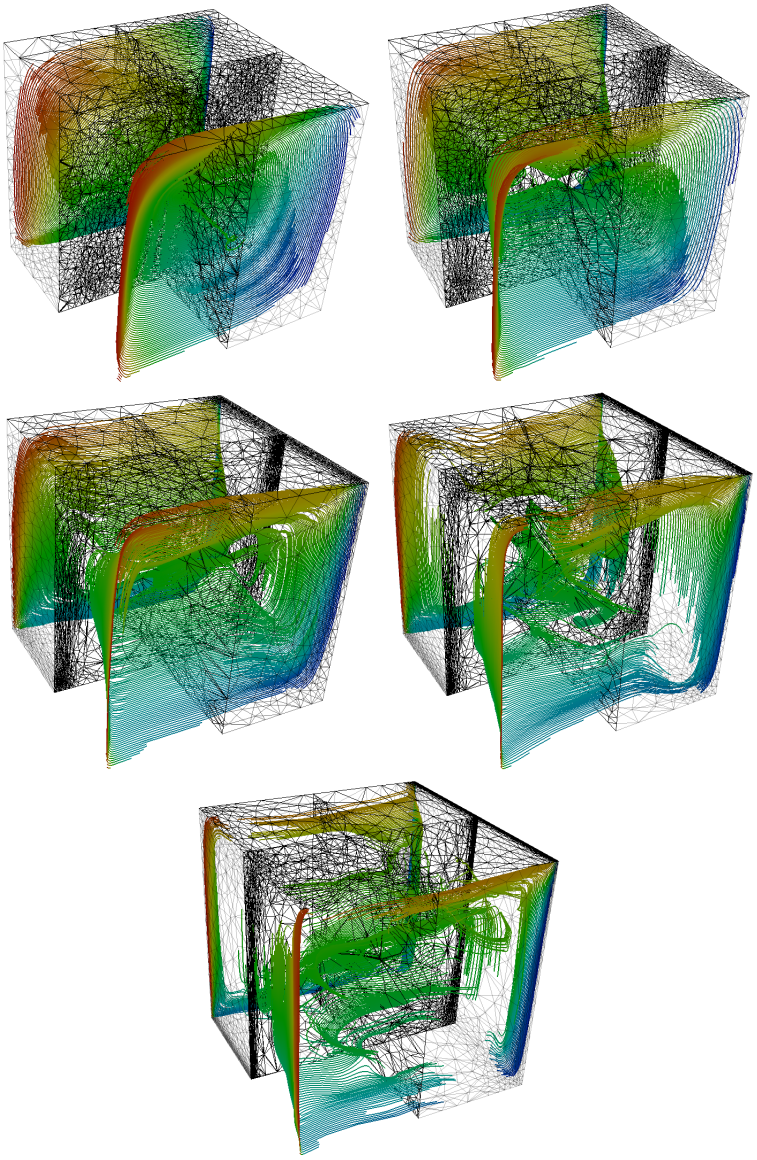


FIGURE 6.24: 3D natural convection streamlines for different Rayleigh numbers : 10^4 (top left), 10^5 (top middle), 10^6 (top right), 10^7 (bottom left), 10^8 (bottom right)

Résumé français

L'objectif de ce travail est de réduire le coût des calculs d'écoulements turbulents couplés aux transferts thermiques. Par conséquent l'utilisation de maillages isotropes n'est pas viable et est exclue naturellement. Le but est donc de capturer les différentes échelles de la physique du problème par une méthode d'adaptation de maillage anisotrope. Une méthode basée sur une estimation d'erreur a posteriori sur les arrêtes du maillage est présentée dans ce chapitre. Cette méthode s'intègre parfaitement dans un contexte d'applications industrielles dans le sens où elle contrôle un nombre fixe de noeuds. L'erreur sur chaque arrête est calculée grâce à une reconstruction du gradient de la solution physique. Cette erreur permet de définir des facteurs d'étirement et de calculer

une nouvelle métrique, menant à un maillage adapté et (potentiellement) fortement anisotrope. Cette méthode d'adaptation est originale car elle considère naturellement plusieurs champs d'adaptation, évitant ainsi des intersections de métriques.

La méthode présentée est comparée à deux méthodes d'intersection de métrique sur un cas de convection forcée 2D. Il en ressort que la méthode prend en compte chaque critère d'adaptation (vitesse, température et level-sets) contrairement aux méthodes d'intersection. En effet le maillage produit par la méthode d'adaptation basée sur les arrêtes est très raffiné et anisotrope vers les couches limites tout en capturant bien les interfaces des objets et la température. Finalement la méthode d'adaptation est mise à l'épreuve sur un cas de convection naturelle 2D. Des courbes de vitesse et de température sont comparées à un cas de référence pour différents critères d'adaptation et différents nombres de Rayleigh. Les résultats montrent qu'adapter le maillage sur la tempéture permet d'obtenir les meilleurs résultats en terme de précision pour ce cas. L'adaptation de maillage anisotrope permet de capturer la solution (vitesse et température) de manière très précise pour un nombre de Rayleigh allant jusqu'à 10^8. Des résultats de convection naturelle 3D sont également présentés dans ce chapitre afin de montrer la capacité de la méthode d'adaptation à produire des maillages fortement anisotropes et l'aptitude des solveurs à capturer parfaitement la solution du problème sur ces éléments très étirés.

Chapter 7

Industrial applications

Numerical prediction of industrial processes such as quenching or heating in industrial furnaces has attracted considerable attention in the past few years. As the processes are complex because of the variety of the phenomena implied and the complicated geometries, it can be difficult and expensive to understand and analyze them through experimental tests. Therefore numerical tools have become essential to understand further these processes, predict the physics and better control them. The thermal history of the temperature into the high alloy steel pieces is essential for their final microstructure, and therefore their quality, in the view of their future use.

In this chapter we assess the performance of the methods presented in the previous chapters. The anisotropic mesh adaptation method as well as the solvers are tested on complex 3D industrial applications. The first case is the cooling of a hat-shaped disc by natural convection. Different mesh adaptation criteria are used in order to analyze the influence on the computed numerical solution. The second case is the heating of a cylindrical ingot in a circular industrial furnace. The results obtained with anisotropic mesh adaptation are compared with computations done with a fixed mesh. The third case is the quenching of a tubular ingot in water. The numerical results of all the cases are compared with experimental data in order to verify the accuracy of the proposed methods.

7.1 Cooling of a hat-shaped disc

In this section we simulate the cooling of a hat-shaped disc by natural convection during $1670s$. The inconel 718 workpiece is initialy hot ($1160^{\circ}C$) and is immersed in a large domain composed of ambient air ($20^{\circ}C$). The VMS and SCPG solvers presented in the previous chapter are used to solve the fluid mechanics and the thermal equations. The Boussinesq approximation is taken into account in order to model the influence of the

temperature on the density. No turbulence model has been considered but the radiative model presented in the previous chapter is used. As in the experimental case, the disc is placed in a wide room, a large cavity is used in the numeical simulation in order to avoid boundary conditions dependancies. Therefore the numerical domain has a dimension of $20m \times 20m \times 20m$ whereas the disc has a radius of $0.25m$ and is $0.1m$ height. Note that this kind of approach is clearly unaffordable with uniform meshes as the number of cells would definitely increase, and the computational time become expensive. The anisotropic mesh adaptation allows dealing with such domains as the mesh will be well adapted in the regions of interest and derefined where the physics does not play a key role as shown in Figure 7.1.

Figure 7.2 shows the temperature profile on the zero iso-value of the level-set of the disc at the end of the computation. One can notice that the mesh is well adapted all around this interface in order to capture accurately the temperature evolution inside and around the workpiece as well as the heat transfers between the disc and the air. There is a clear temperature difference between the top and the bottom part of the workpiece (around $200^{\circ}C$). That means that the bottom part cools down faster than the upper part. Indeed, the bottom surface is larger and exposed to the surrounding calm air, whereas at the top, the heated air interacts all along the simulation as shown in Figure 7.3. Note that the induced flow goes higher as the surrounding of the disc is heated. We also point out that the mesh follows the temperature convection inside the domain. Figure 7.4 presents some snapshots for the temperature profile of the workpiece for different times. We can see how the solid cools down during the computation while the interface remains accurately captured.

In order to validate the used methods (Immersed Volume Method with anisotropic mesh adaptation and stabilized finite elements methods), we compare the numerical results of different simulations with experimental data. The computations have been all done with dynamic anisotropic mesh adaptation. Two case were considered; in the first, we adapt the mesh on both the temperature and the level-set function, while in the second only the temperature is considered. We have not adapted the mesh on the velocity field as the latter is clearly coupled to the temperature field in a natural convection way. For both cases, different computations have been simulated with a varying number of elements: $100,000$, $200,000$ and $400,000$. This leads to a total of six test cases. The numerical results are depicted in Figure 7.5. The first point to highlight is that the numerical results are in general in good accordance with the experimental data. At the end of the simulation, all the computations drift from the experimental data by less than 10%. Adapting the mesh on the level-set of the solid leads to a slower cooling. This was expected as the interface between the air and the disc remains well captured by the mesh. In fact, the accuracy of the interface is crucial in order to have proper mixing laws and thus model well the transfers between the fluid and the solid. A coarse interface leads to more diffusive results as can be seen on the plots of Figure 7.5.

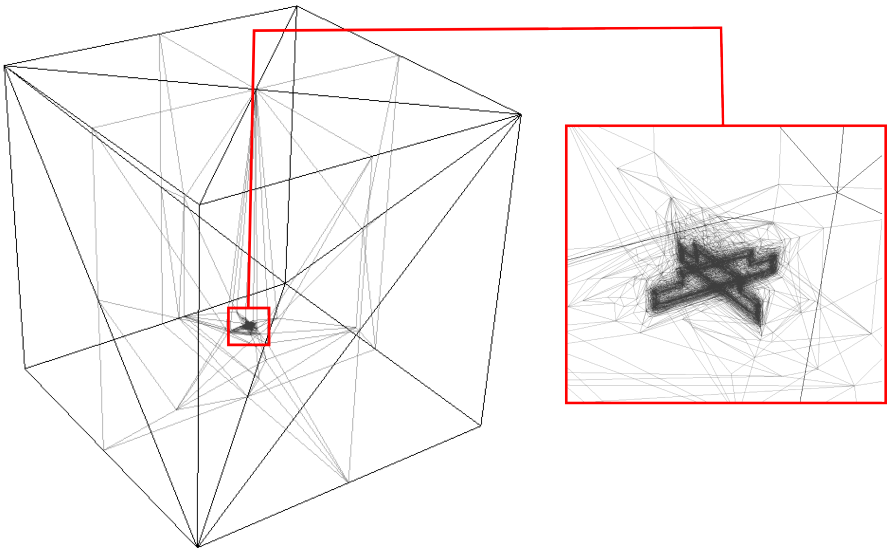


FIGURE 7.1: Cuts of the mesh in the computational domain and zoom on the adapted zone

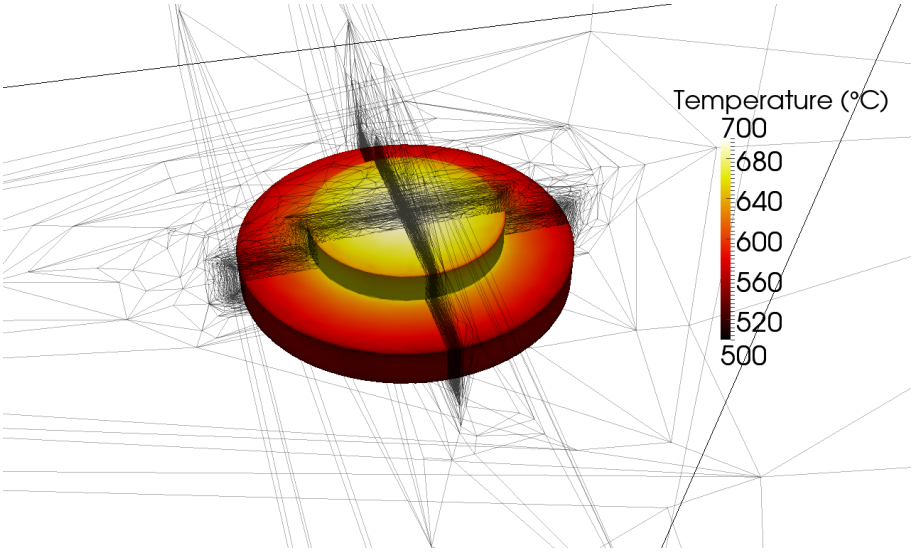


FIGURE 7.2: Zoom on the zero iso-value of the workpiece level-set function, temperature profile and adapted mesh at the end of the computation ($t = 1670s$)

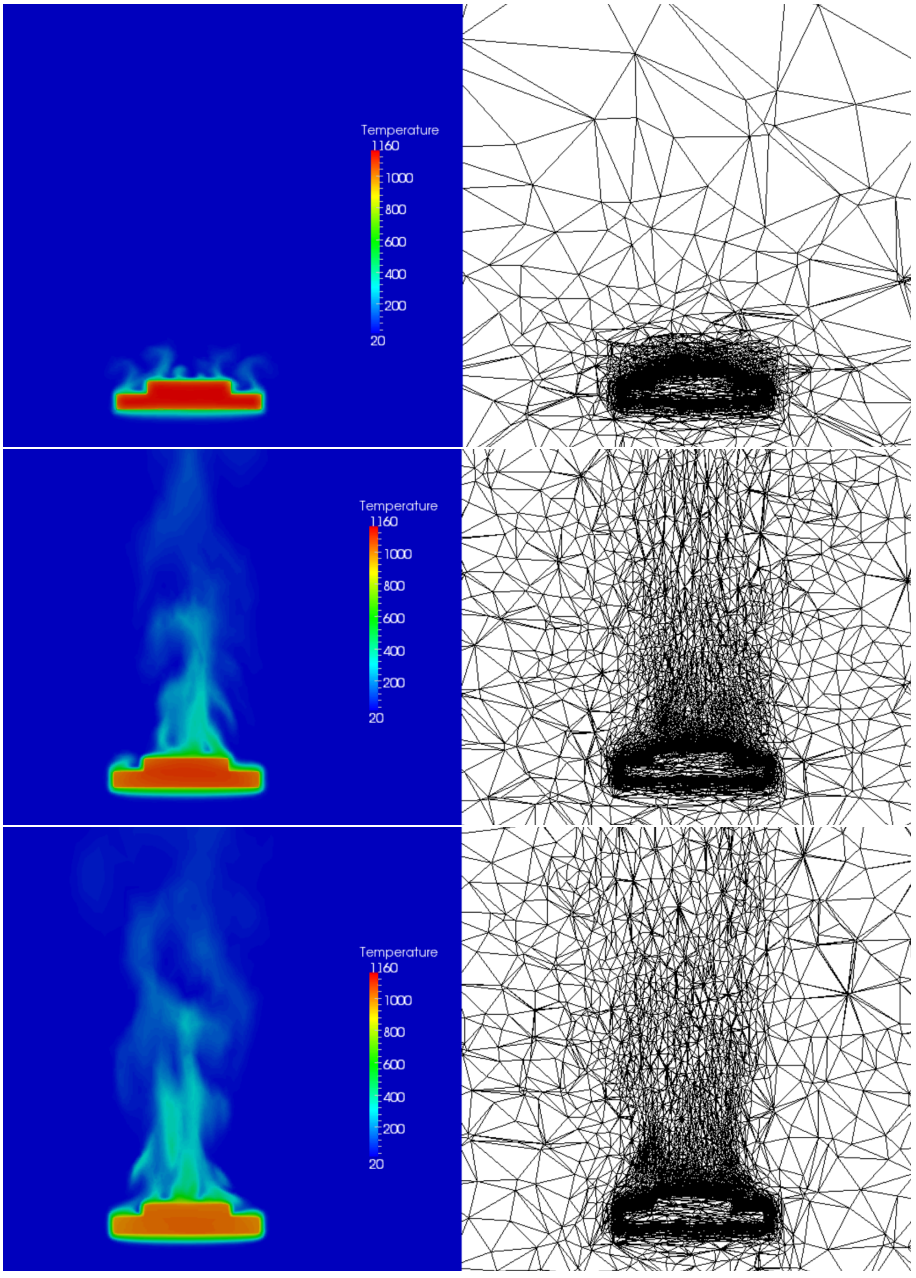


FIGURE 7.3: Cuts in the plane for the temperature profile and the adapted mesh at different times: $t = 35s$ (top), $t = 670s$ (middle) and $t = 1670s$ (bottom)

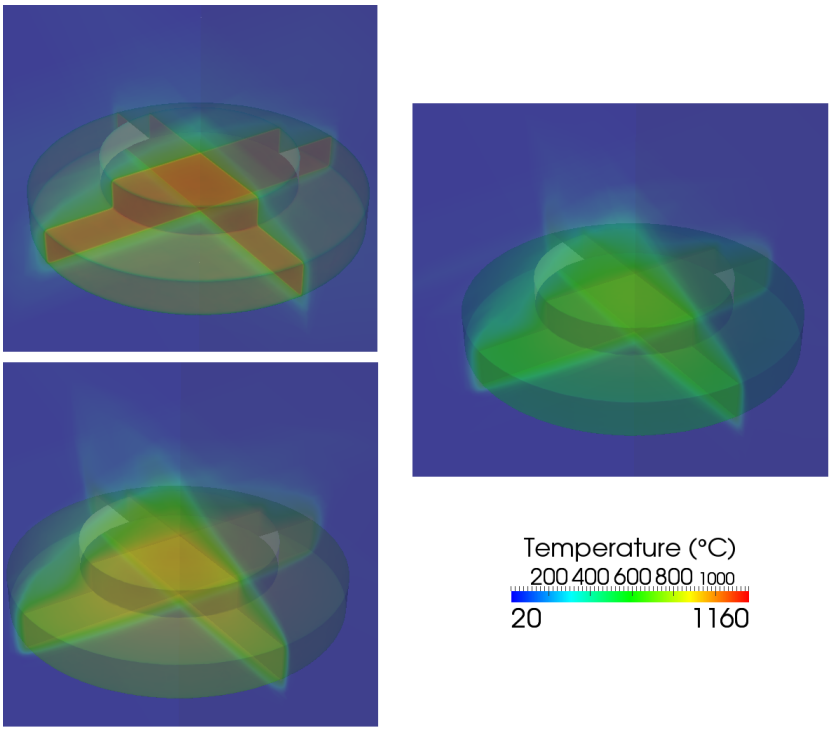


FIGURE 7.4: Zoom on the temperature profile at different times: $t = 35s$ (top left), $t = 670s$ (bottom left) and $t = 1670s$ (right)

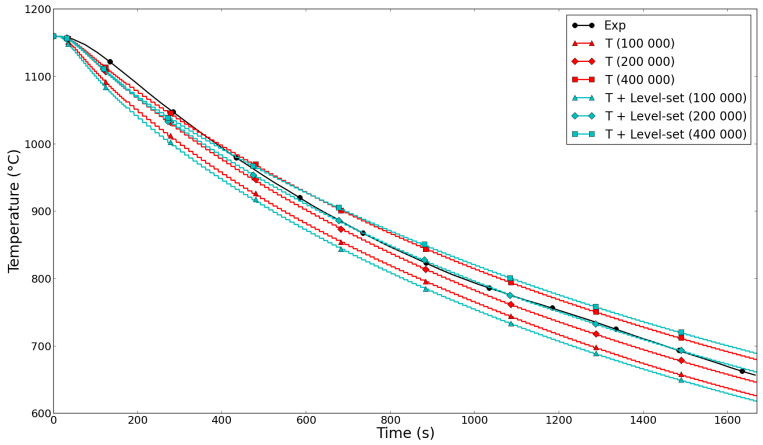


FIGURE 7.5: Temperature evolution inside the disc during the computation. The plots named $T + (n)$ correspond to the computations with a mesh adapted on the temperature and made of n elements. The curves named $T + Level - set(n)$ correspond to the computations with a mesh adapted on both the temperature and the level-set, and made of n elements

7.2 Heating in an industrial furnace

Heating in furnaces is a process widely used in the steel industry and particularly for thermal treatment of metals. In these furnaces the temperature can be higher than $1200^{\circ}C$. Therefore heat transfers are mainly done by radiation of the flame and the gas from combustion. The thermal history inside these furnaces is essential for the industrials in order the get workpieces with the desired mechanical properties. In this section, we consider the heating of a cylindrical ingot in a circular industrial furnace. The initially cold ingot is positionned at the bottom center of the furnace. Air is burned and injected through four inlets and is evacuated through the outlet at the top wall of the furnace. The mesh is initially adapted on the level-set of the ingot and also on the inlets and the outlet. The configuration as well as the initial adapted mesh can be seen in Figure 7.6. The ingot is heated by the burners during three hours. At the beginning of the simulation, the four burners are turned on and at time $t = 840s$, the burners $B3$ and $B4$ are turned off. Therefore only the burners $B1$ and $B2$ keep blowing hot air. No slip and adiabatic boundary conditions are applied at the walls of the furnace. The ingot is initially at $23^{\circ}C$ whereas the inside of the furnace is preheated at certain temperature. The burned air is injected at a higher temperature. We do not mention these values as well as the velocity of the air blown by the burners for confidential reasons. During the computation the mesh is adapted dynamically on the temperature, the level-set of the ingot and the velocity field.

Figure 7.7 presents cuts of the temperature at the level of the burners at different times. We can clearly see that the four burners are activated and then two of them are turned off. Similar cuts for the velocity magnitude are provided in Figure 7.8. We can see the difference in the flow before and after the burners have been turned off. Before, the rotation of the flow is settled near the wall and also inside the furnace. This is confirmed in Figure 7.10. In fact burners $B3$ and $B4$ deviate the jets coming out from burners $B1$ and $B2$, thus breaking these jets and inducing the rotated motion of the flow. Whereas when the burners $B3$ and $B4$ are turned off, the jets of the two other burners go straight until hitting the vertical wall. Therefore the flow still rotates as the furnace is circular, but is more confined near the wall. Figures 7.9 and 7.11 show also the mesh before and after turning off the two burners as well as in the vicinity of the ingot. We can see that the mesh captures well the air-solid interface but also follows these changes in the boundary conditions. Indeed the mesh is well adapted on the four burners at the beginning and then concentrates the elements on the two remaining jets. This is a clear advantage of the dynamic mesh adaptation. As the burners are turned off, there is no need of keeping a fine mesh in their vicinity. Therefore the method automatically changes the location of the elements concentration to follow the two remaing jets. We can also notice that the mesh is also well adapted at all the boundary layers of the domain. Figure 7.12 compares the temperature evolution of ingot at different positions. Four sensors have been placed: three at the mid-height of the ingot, at its center, its

surface and in-between, and one at the center of its top surface. As expected the solid is heated faster at its surface and slower at its center. We can also notice that it is also heated faster at the top surface. This can be explained by the fact that the burners are in the upper part of the furnace.

In order to test the dynamic mesh adaptation efficiency, we have run three computations. The first one takes into account a dynamic mesh adaptation on the temperature, the level-set of the ingot and the velocity field as well. This case uses approximately $500,000$ elements. The two other simulations have been done with a fixed uniform mesh of $250,000$ and $500,000$ respectively. The numerical results of each computation are compared with the experimental data and depicted in Figures 7.13, 7.14, 7.15 and 7.16. First, we can see that the temperature given by all the numerical results do not fit the experimental data. We explain this by the fact that we did not use a propper model to take into account accurately the gas combustion and radiation. Also more elements are needed in order to get more accurate results and capture better the flow and the interface of the ingot. Finally the heat transfer between the air outside of the furnace and the walls should be taken into account properly. However the results provided by the adapted case are better than the results of the fixed mesh cases. The rise in temperature follows better the expected profile of the experimental data. At the beginning the temperature increases progressively whereas the fixed mesh cases provide a brutal rise. Then the adapted case follows better the continuous heat of the ingot whereas the fixed mesh cases stagnate. We can also notice that both fixed meshes are not fine enough to capture accurately the initial temperature profile along the inteface of the ingot. Indeed in Figures 7.15 and 7.16, the temperature at the beginning of the fixed mesh computations is not at $23^{\circ}C$. Both sensors are close to the interface, therefore the meshes are not fine enough to provide accurate mixing laws and thus proper air-solid interface. All the computations have been run on 40 2.4Ghz Opteron cores. Table 7.1 shows the computational time in hours of the three cases. We point out that with approximately the same number of elements, the adapted case is faster than the fixed mesh case using 500,000 elements. Therefore we can conclude that in one hand the computational time is spent to adapt the mesh, but in the other hand it leads to a faster convergence of the solvers.

TABLE 7.1: Computational time in hours of the three cases after $10,800s$

Adapted	Fixed (250,000)	Fixed (500,000)
185	70	201

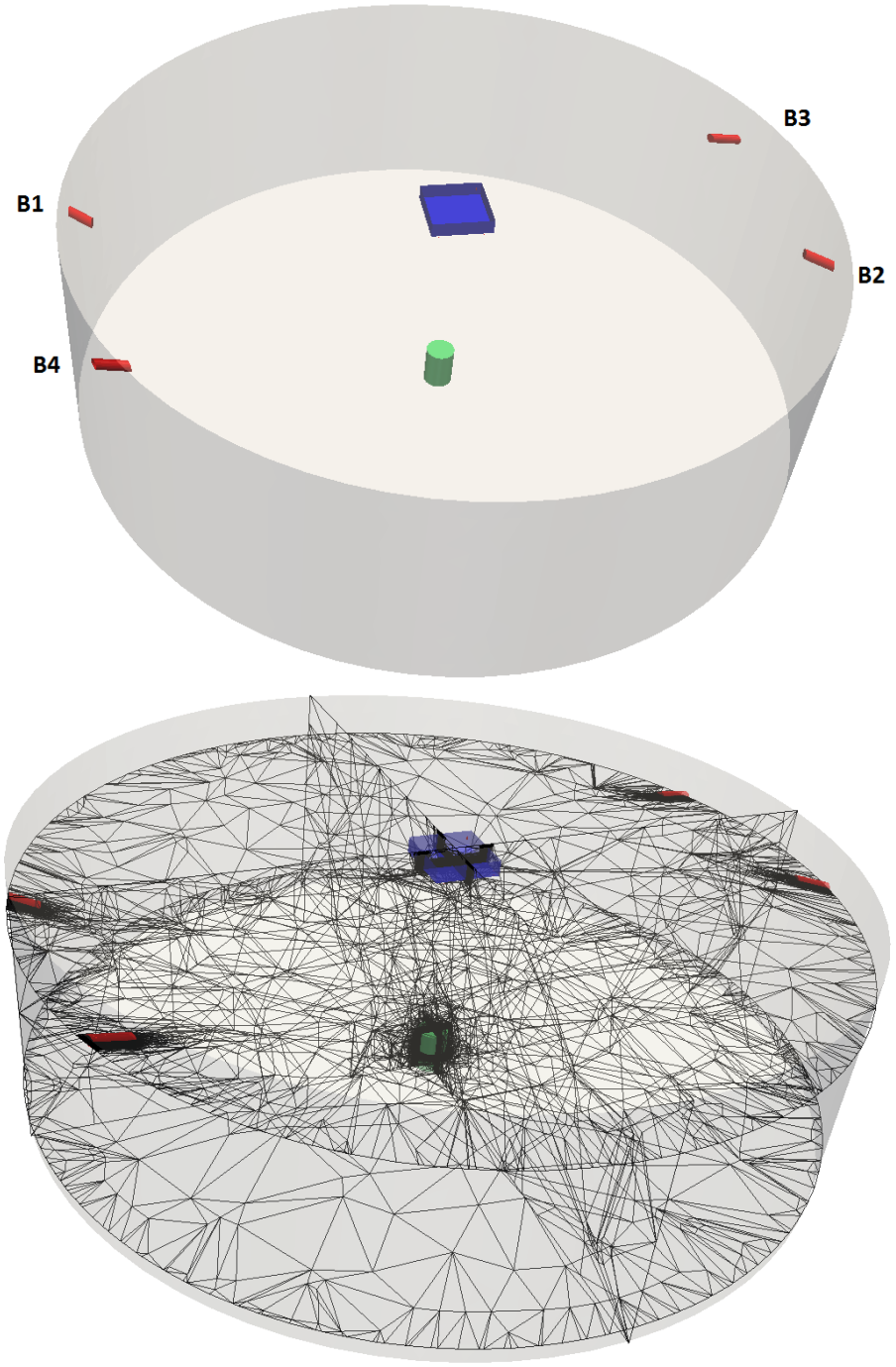


FIGURE 7.6: Configuration of the furnace (top): the ingot (green) is placed at the bottom center, four burners (red) blow hot air which is evacuated through the outlet (blue) on the top wall. Initial adapted mesh (bottom)

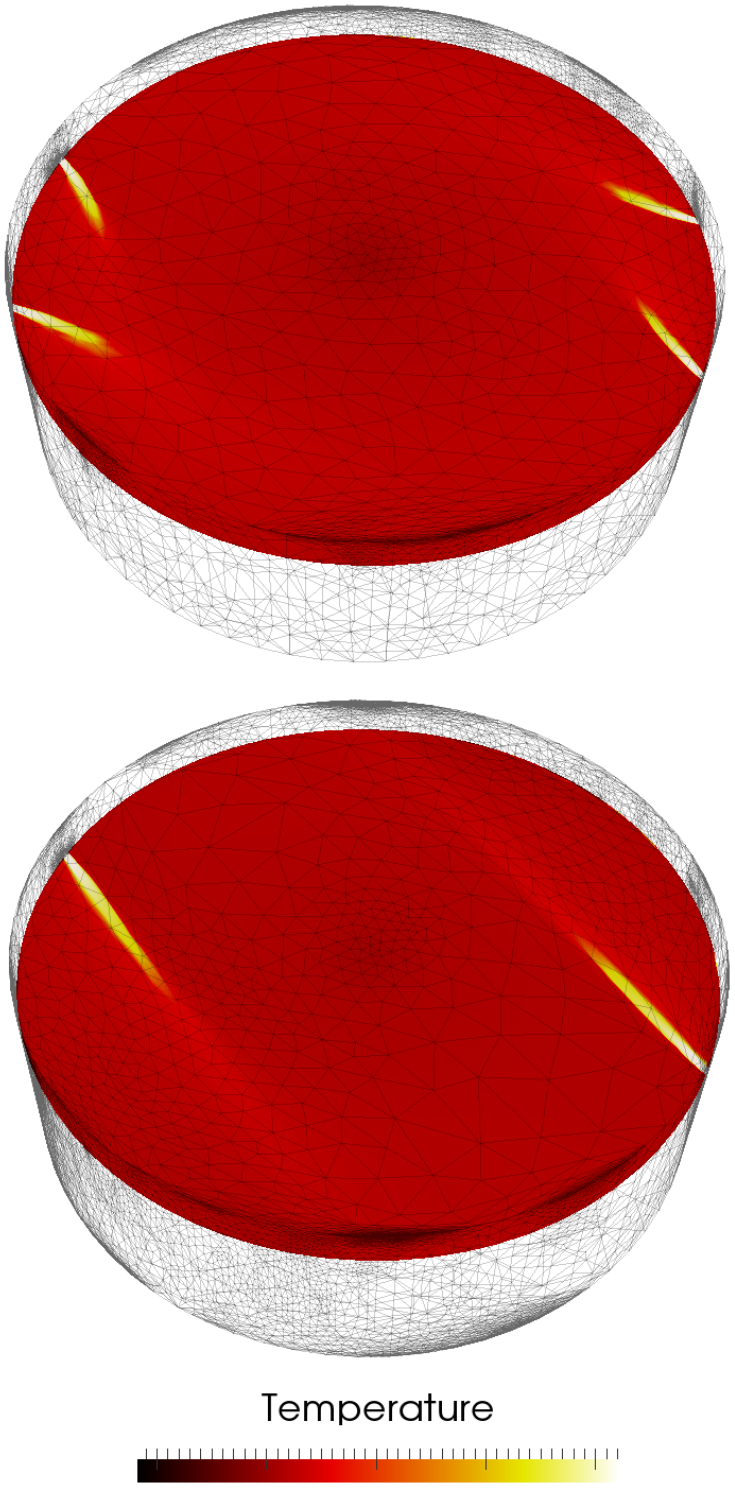


FIGURE 7.7: Cuts of the temperature at the burners level at different times: $t = 829s$ (top) and $t = 10800s$ (bottom)

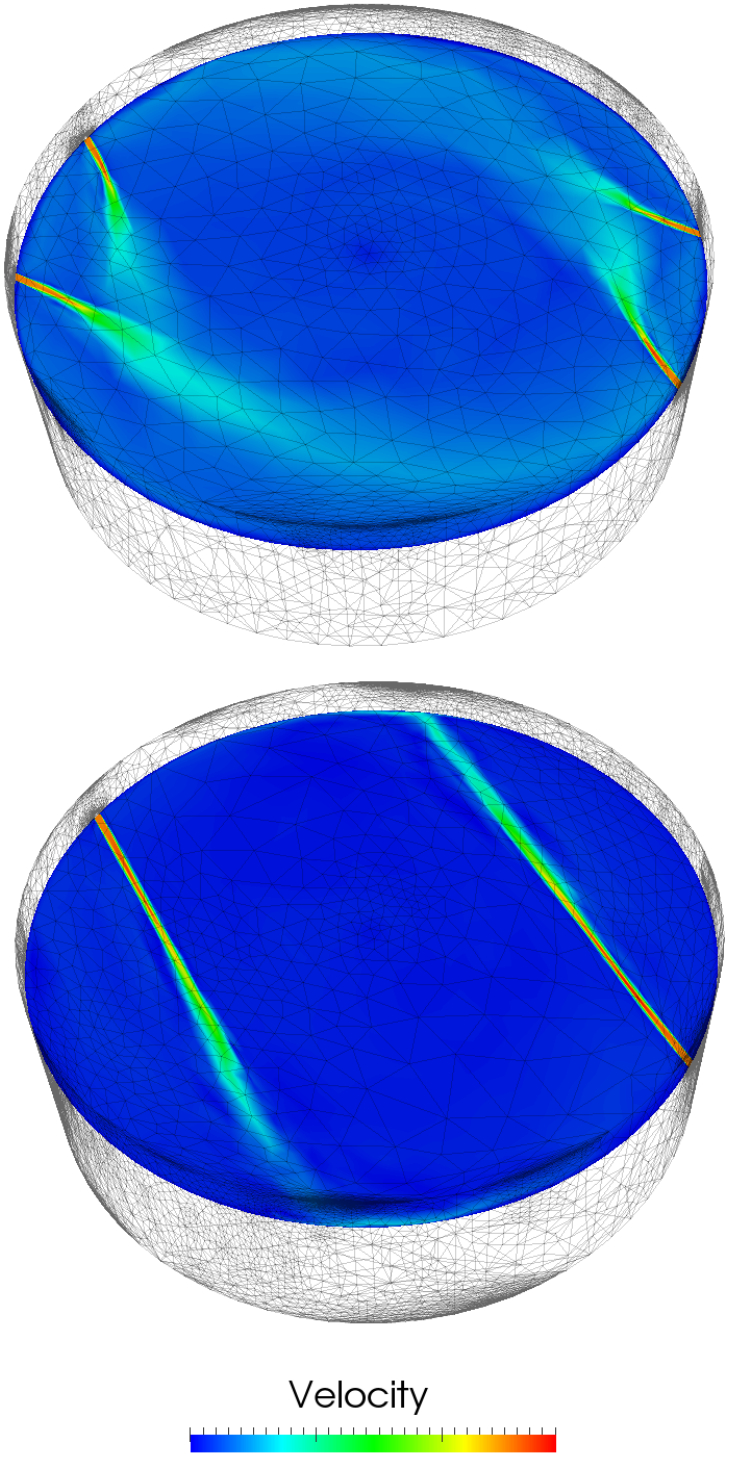


FIGURE 7.8: Cuts of the velocity at the burners level at different times: $t = 829s$ (top) and $t = 10800s$ (bottom)

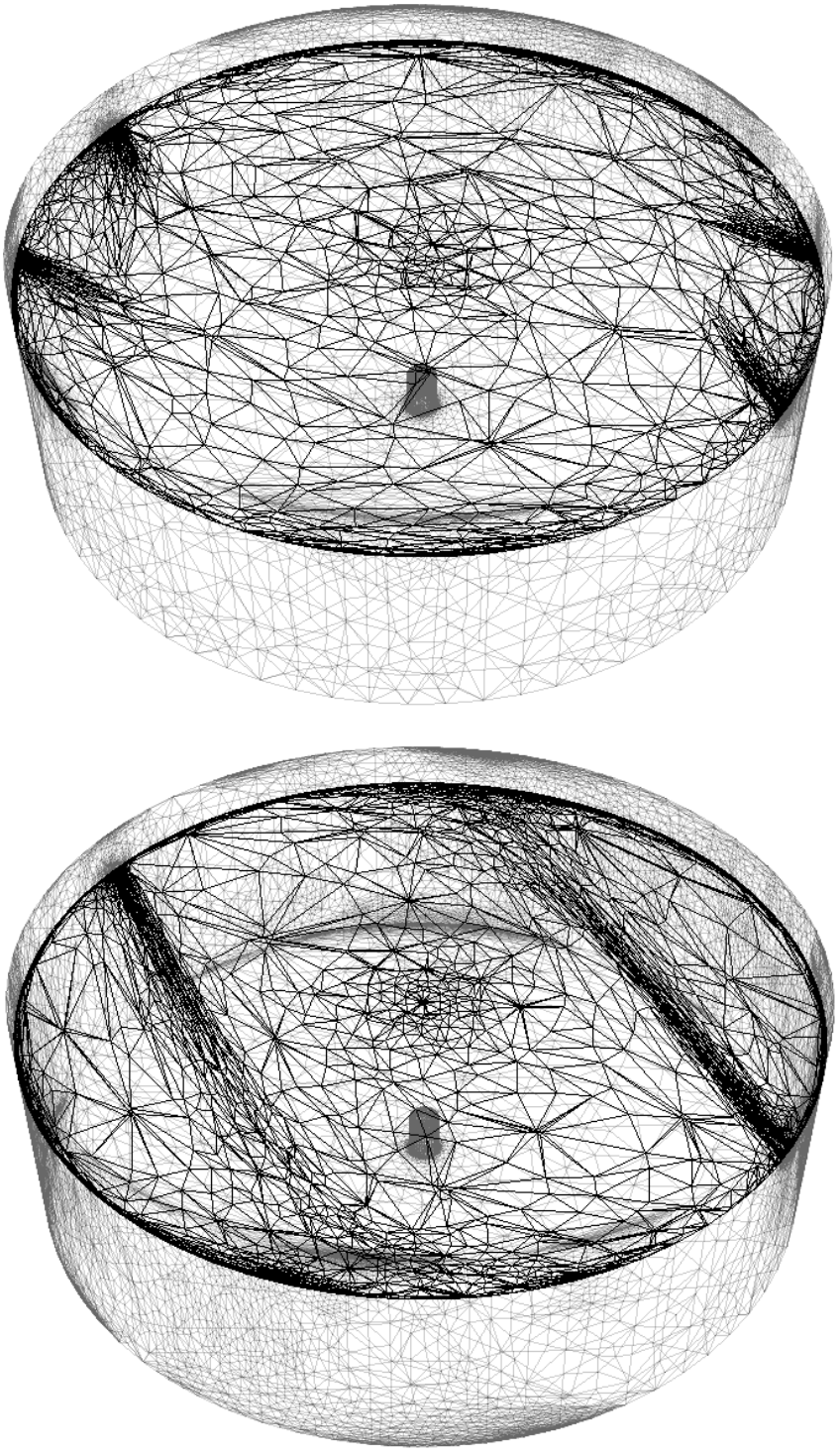


FIGURE 7.9: Cuts of the mesh at the burners level at different times: $t = 829s$ (top) and $t = 10800s$ (bottom)

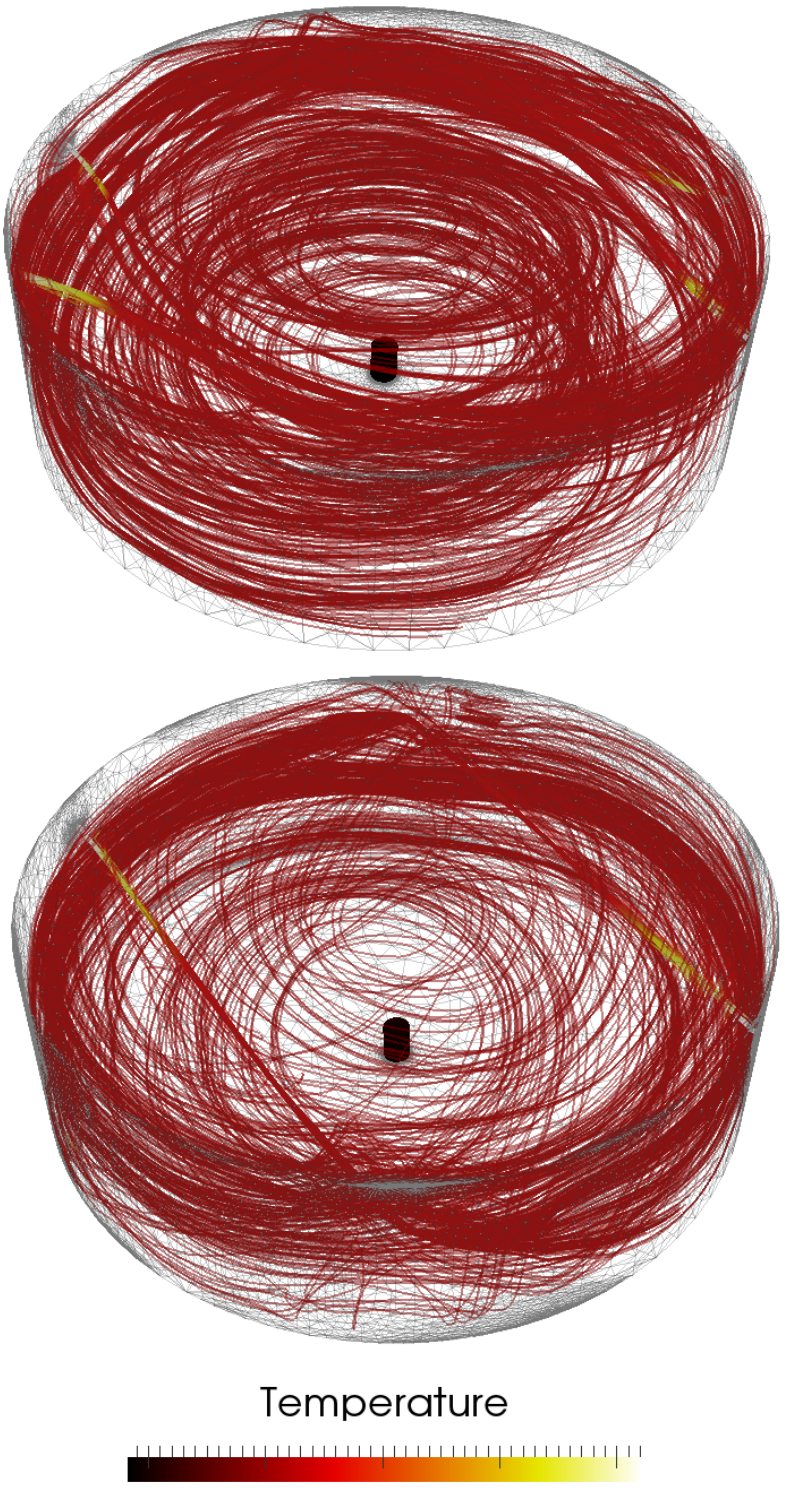


FIGURE 7.10: Streamlines at different times: $t = 829s$ (top) and $t = 10800s$ (bottom)

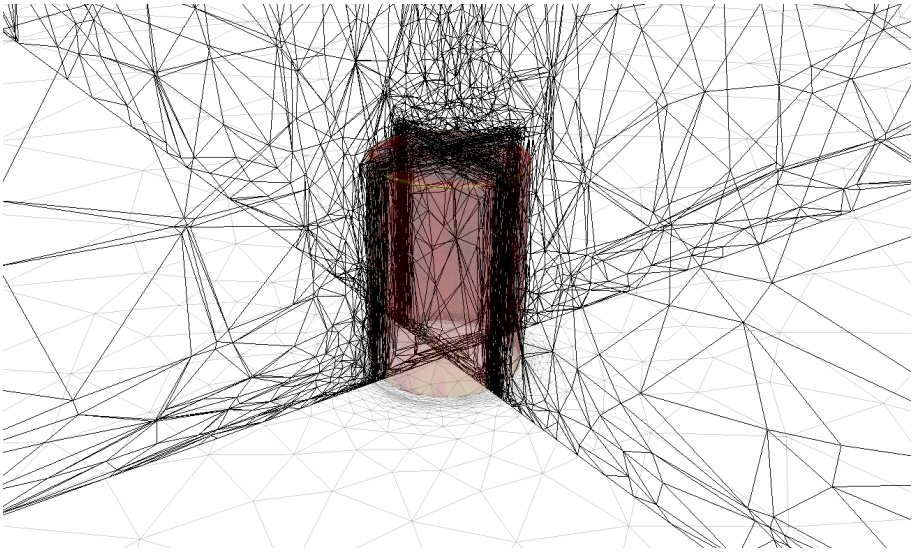


FIGURE 7.11: Zoom on the ingot and the adapted mesh inside the furnace

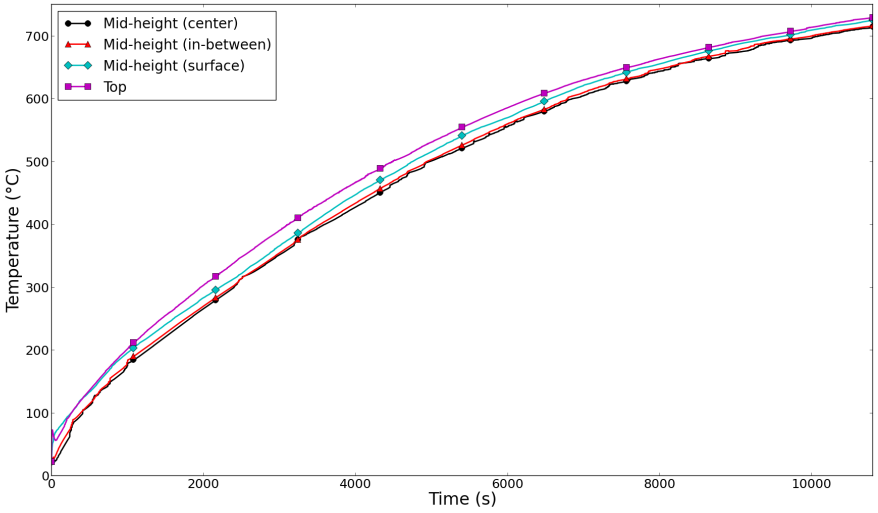


FIGURE 7.12: Temperature evolution at different positions in the ingot

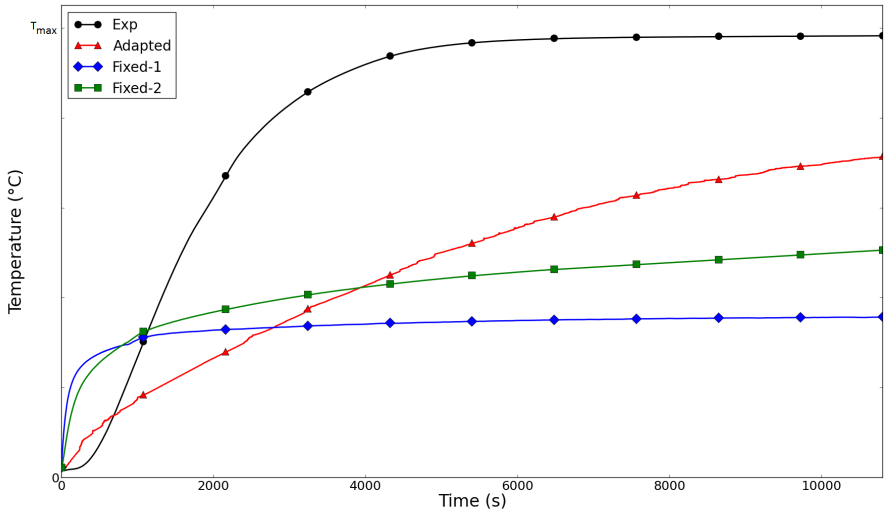


FIGURE 7.13: Temperature evolution in the mid-height of the ingot at the center

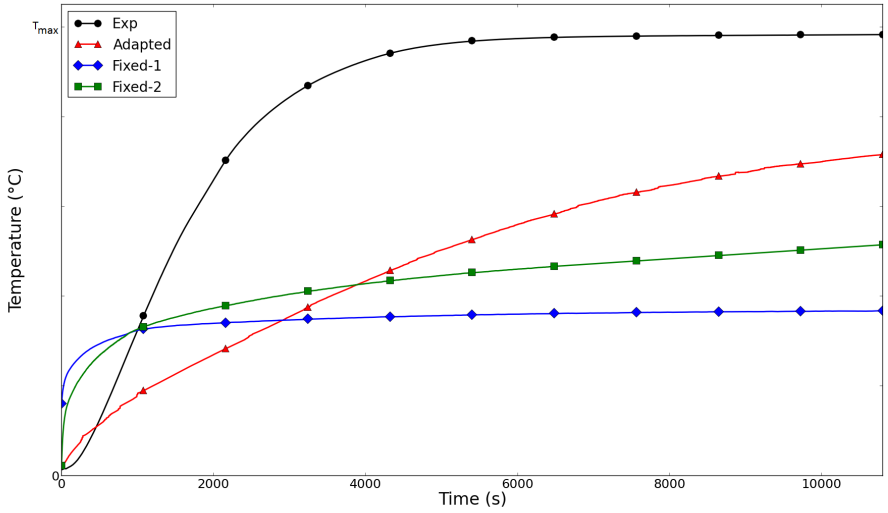


FIGURE 7.14: Temperature evolution in the mid-height of the ingot at the in-between

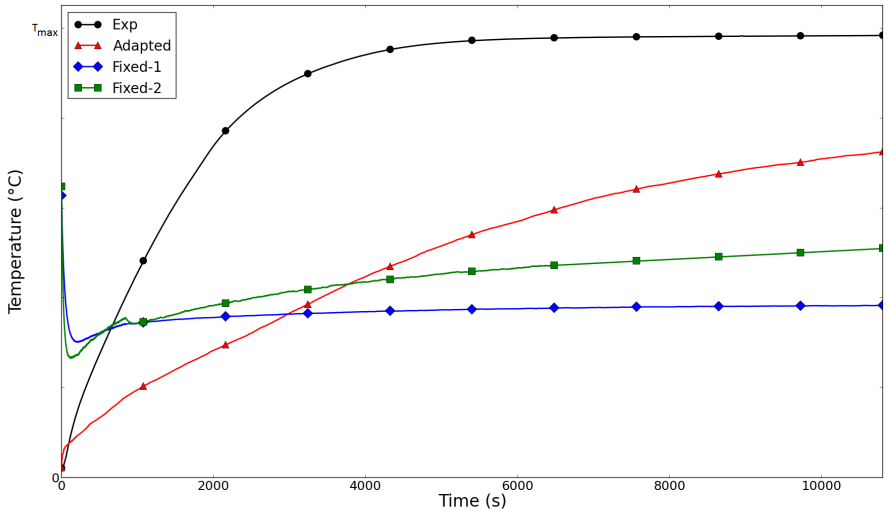


FIGURE 7.15: Temperature evolution in the mid-height of the ingot at the surface

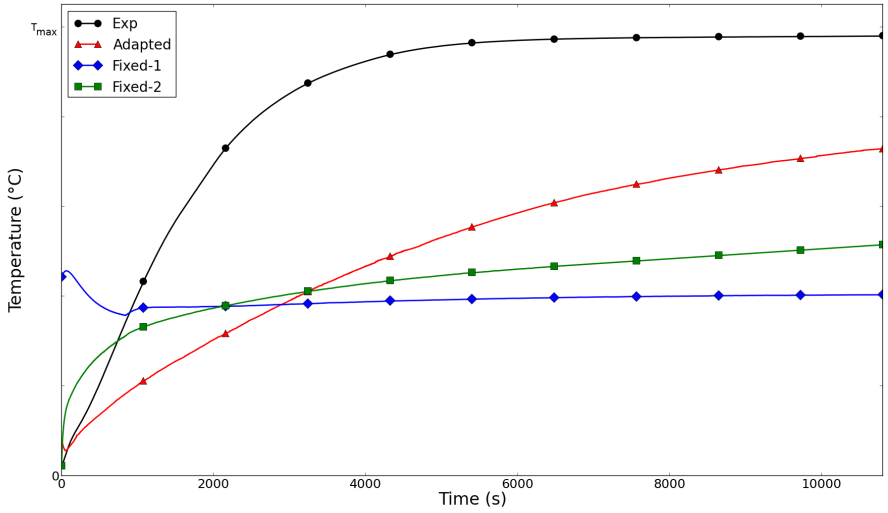


FIGURE 7.16: Temperature evolution in the center of the top surface of the ingot

7.3 Quenching in water

Quenching is a thermal treatment used by industrials to get certain mechanical properties. Usually this cooling process improves the hardness of the workpiece and its behaviour concerning fracture.

In this section we consider the quenching of a tubular ingot in water. The process is divided in two steps. First, water at ambiant temperature is injected at the bottom of the cavity. Eight turbines also induce a flow. The fluid is evacuated through an outlet at the top of the vertical wall. In order to make the flow stabilized, this phase is performed during 20 minutes. Second, the workpiece is added in the cavity. The workpiece is initially hot and the inlet keep injecting ambiant water while the turbines are still turned on. The workpiece cools down during 20 minutes. No slip boundary conditions are used at the walls. Heat transfers with the outside of the cavity are taken into account by considering a heat flux of type Fourier at the top wall and imposing a constant heat flux at the other walls. Figure 7.17 shows the configuration of the first step. The mesh is initially adapted on the level-set functions of all the turbines, as well as the inlet and the outlet. Both the velocity field and all these level-set functions are considered as mesh adaptation criteria during the computation. Around $500,000$ elements have been used for this simulation.

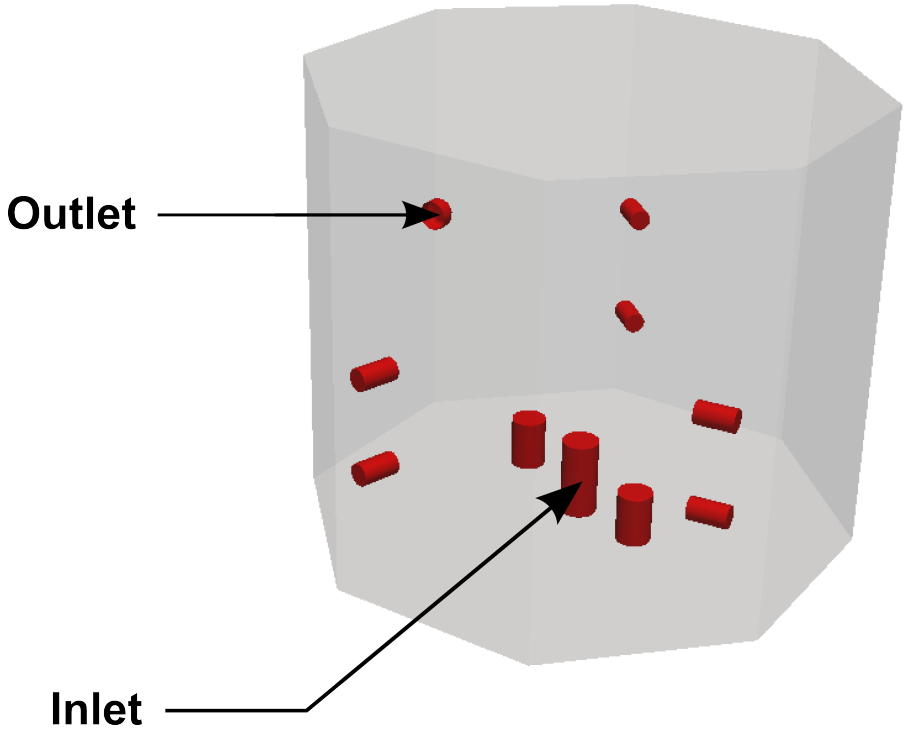


FIGURE 7.17: Configuration of the first step: there are 8 turbines inside the cavity as well as an inlet and an outlet

Figures 7.18 and 7.19 show the streamlines of the flow and cuts of the velocity vectors at the end of the first step of the simulation. Although the flow is clearly chaotic, we can still distinguish a rotative motion. Of course this is due to the orientation of the six top turbines. Figure 7.20 presents the adapted mesh at the end of the computation. We can notice that the mesh is well adapted at the level-set functions interfaces, but also follows the flow, especially at the outlets of the turbines and at the boundary layers. In fact the purpose of this first step of the process is to have a stabilized flow, and use the latter as an input for the second step.

In the second step, the hot tubular worpiece is added. Again, the values of the temperature and the velocity are not provided for confidential reasons. The same number of elements has been used here and the temperature field as well as the level-set are added as a criterion for the mesh adaptation. Figure 7.21 shows the new configuration, while Figures 7.22 and 7.23 present the adapted mesh and the streamlines at the end of the computation. In order to compare the numerical results in terms of temperature evolution with the experimental data, five sensors have been placed in the workpiece. The positions of the sensors can be seen in Figure 7.24. We can see that different strategical positions have been chosen for the analysis.

Figure 7.25 compares the temperature evolution during the second step between all the sensors. As expected the center of the workpiece is the slowest part to cool down. Indeed the sensors placed near the interior or exterior surface show that the temperature decreases faster at these positions. We can also compare the temperature evolution of the two sensors placed at the interior surface (sensors 3 and 5). The two sensors are placed at opposite sides in the workpiece, and the temperature at sensor 3 clearly drops faster. Finally we compare the numerical results obtained for all the sensors with experimental data in Figures 7.26 and 7.27. The numerical results are not in good accordance with the experimental data. But this can explained by the fact that we did not use any model for boiling and phase change, which plays a key role in this kind of process. Indeed when quenching a workpiece at high temperature in water, vapor films film settle around the solid, slowing down the cooling of the solid. Finally, we point out that the computation has taken 4.5 hours on 100 2.4Ghz Opteron cores.

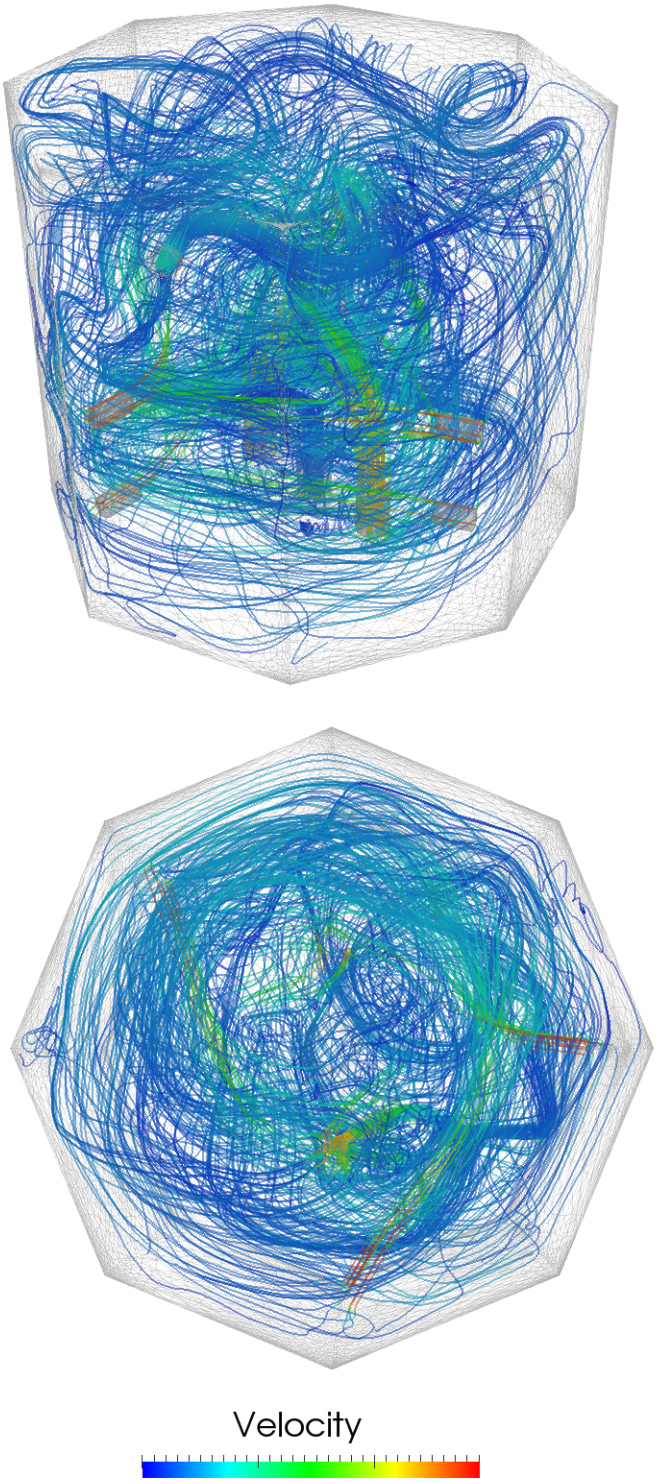


FIGURE 7.18: Streamlines of the flow inside the cavity at the end of the first step computation: isometric view (top), and top view (bottom)

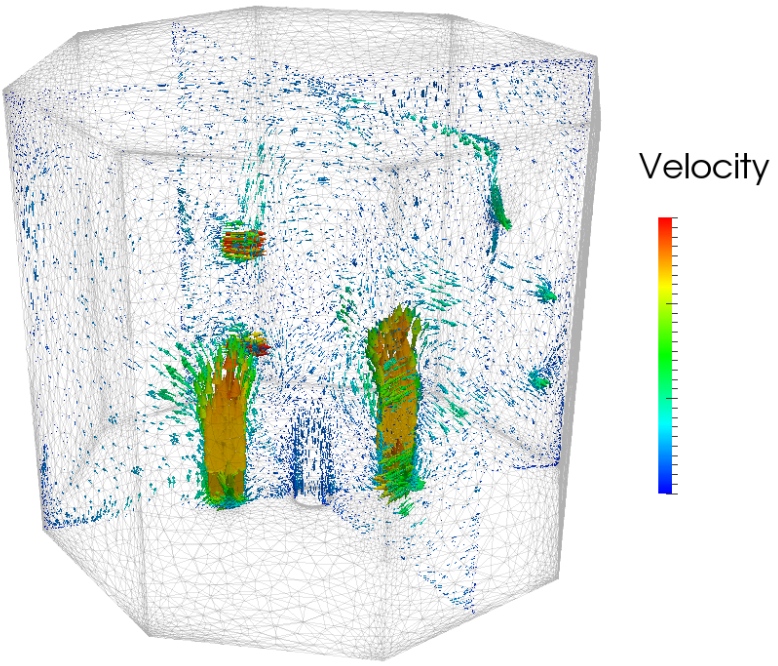


FIGURE 7.19: Cuts of the velocity vectors at the end of the first step computation

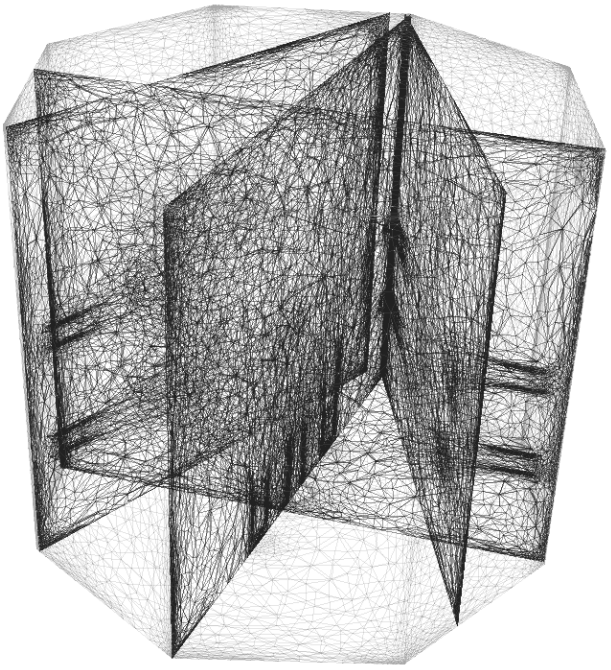


FIGURE 7.20: Adapted mesh at the end of the first step computation

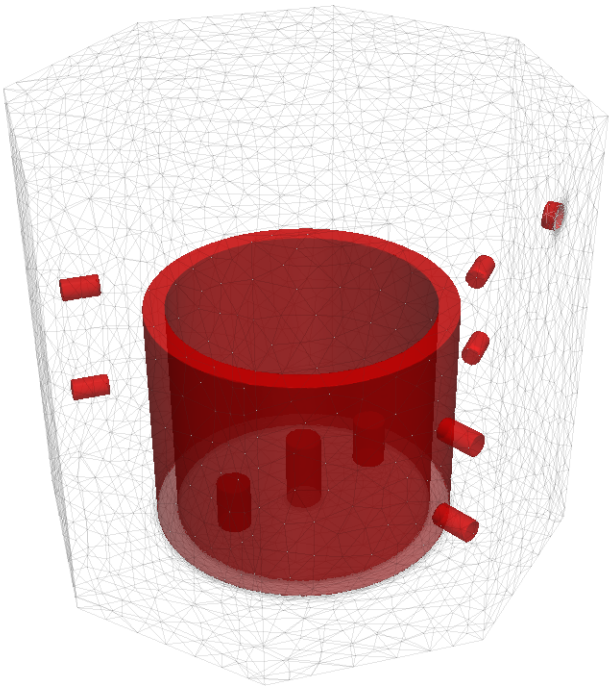


FIGURE 7.21: Configuration of the second step: the tubular workpiece has been added

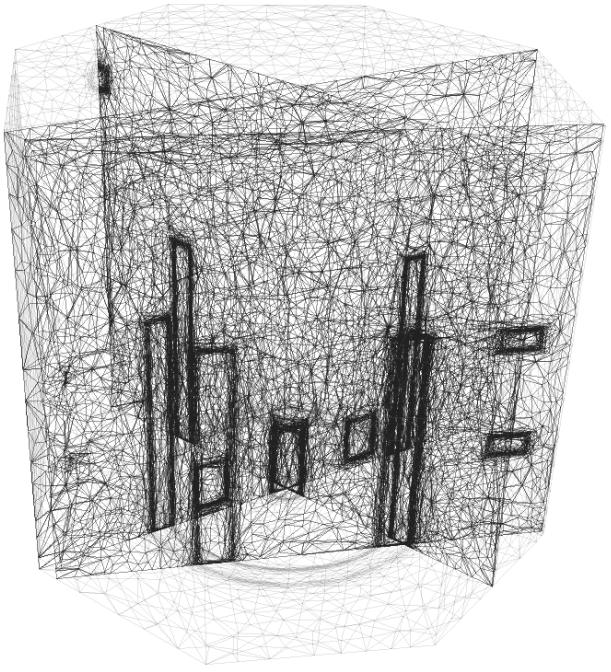


FIGURE 7.22: Adapted mesh at the end of the second step computation

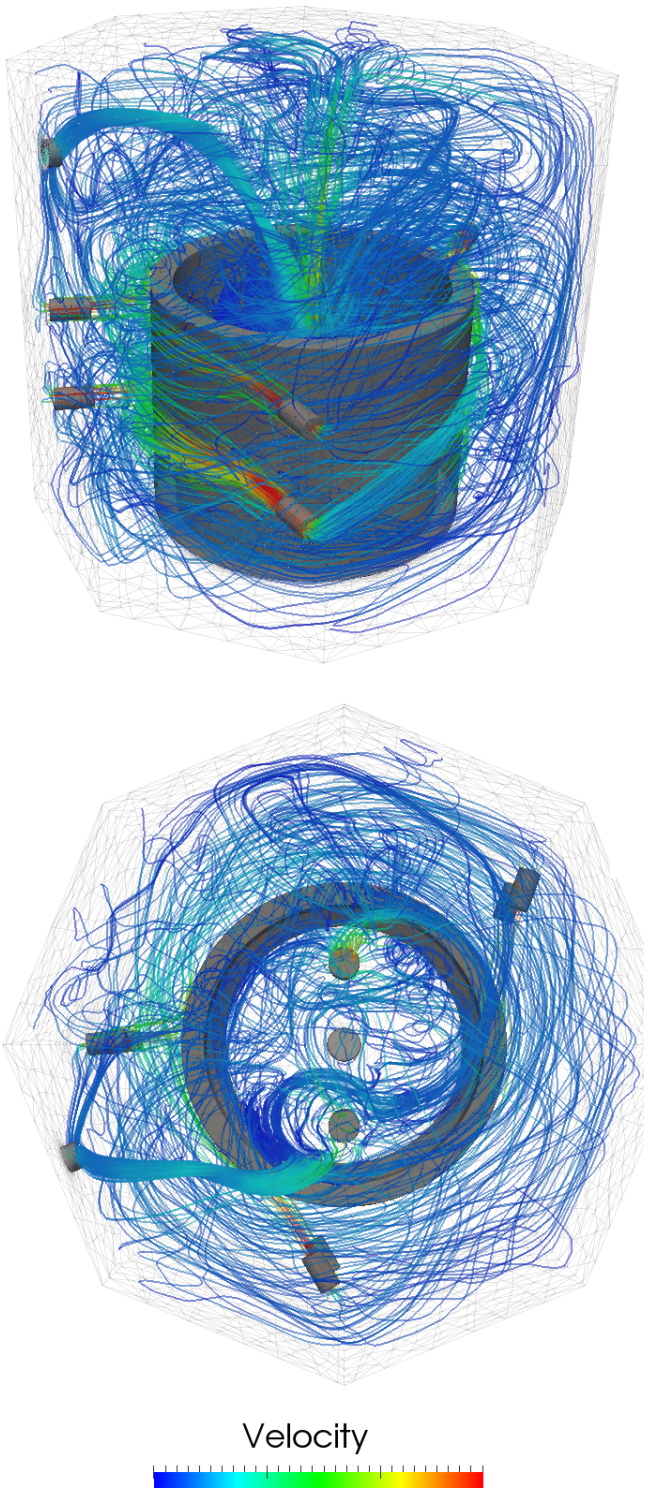


FIGURE 7.23: Streamlines of the flow inside the cavity at the end of the second step computation: isometric view (top), and top view (bottom)

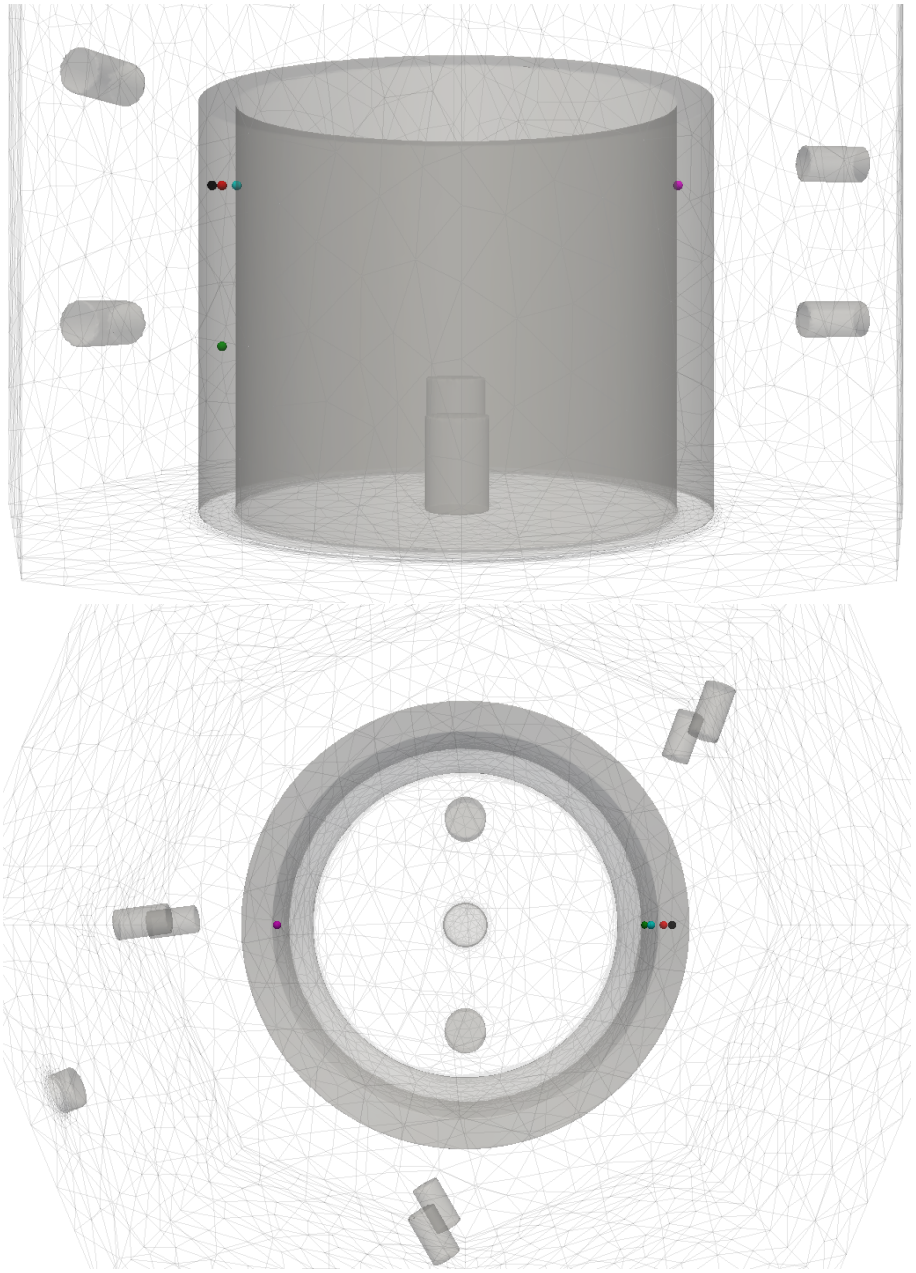


FIGURE 7.24: Position of the five sensors in the workpiece: sensor 1 (black), 2 (red), 3 (blue), 4 (green) and 5 (magenta). Isometric view (top) and top view (bottom)

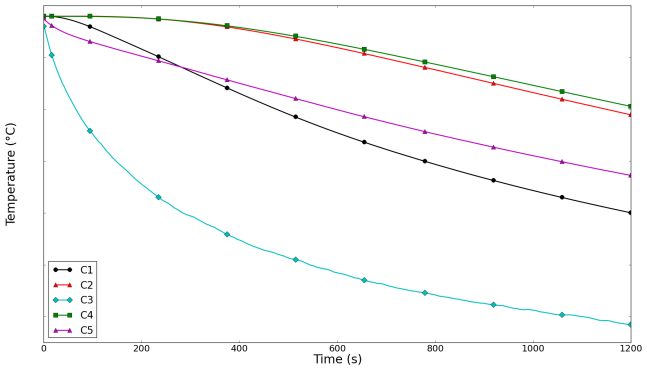


FIGURE 7.25: Comparison of the temperature evolution during the second step between all the sensors

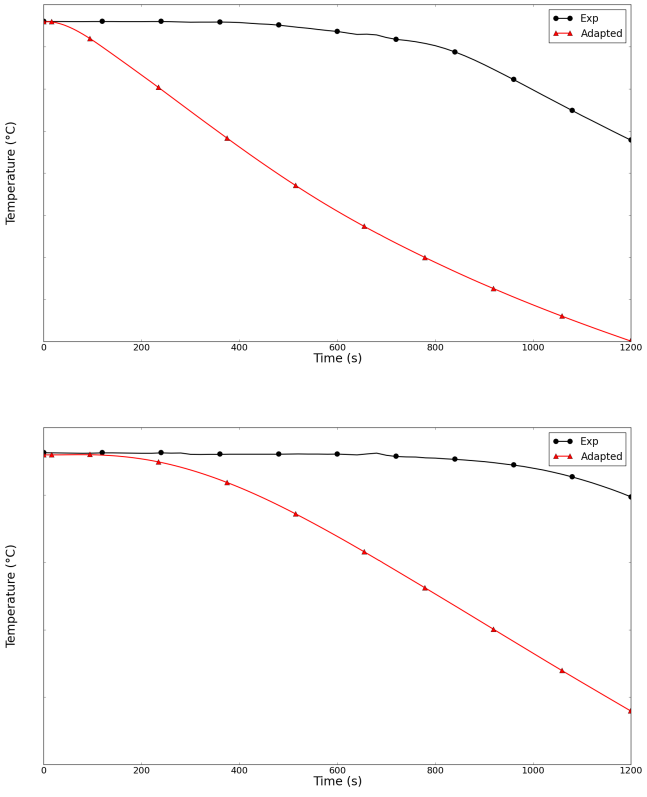


FIGURE 7.26: Comparison with experimental data of the temperature evolution during the second step in sensor 1 (top), 2 (bottom)

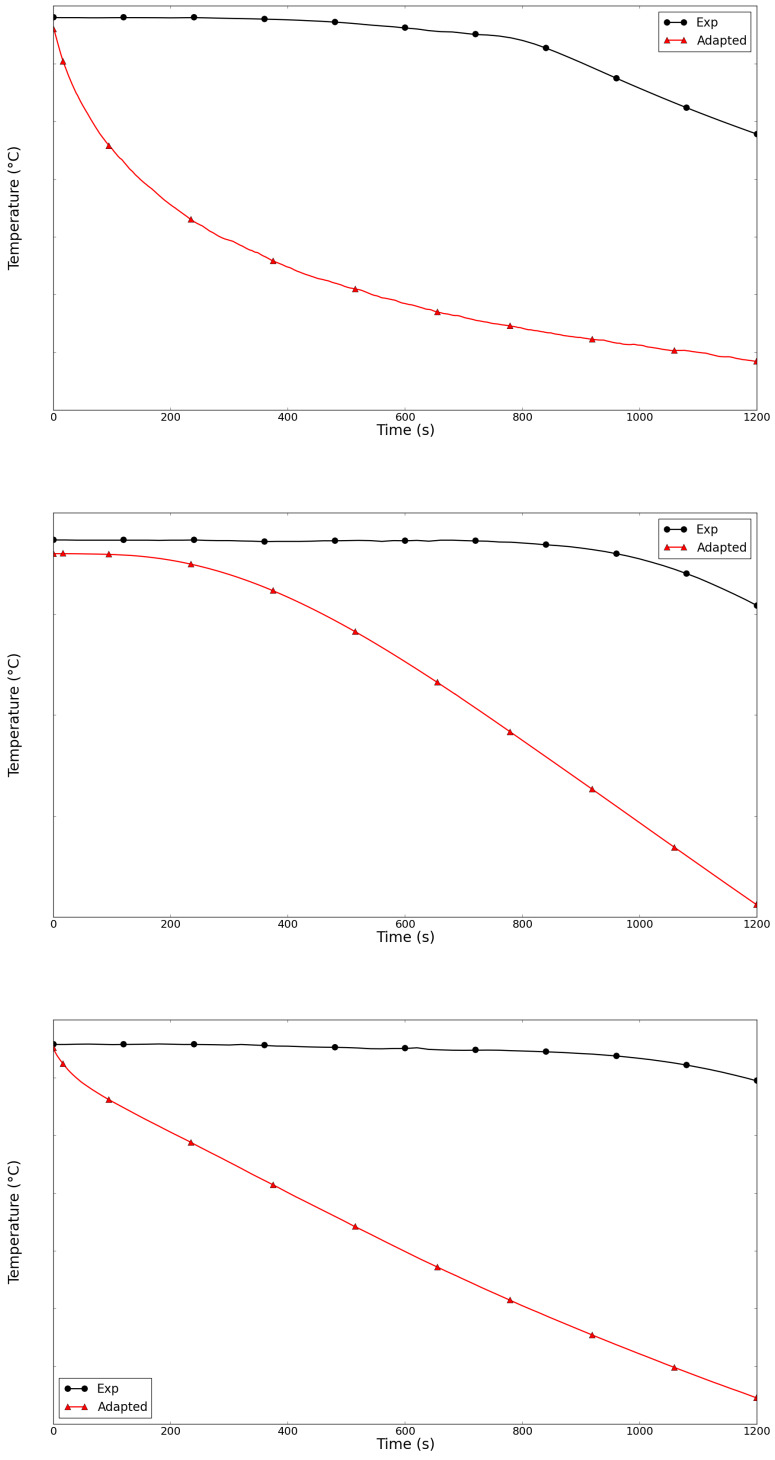


FIGURE 7.27: Comparison with experimental data of the temperature evolution during the second step in sensor 3 (top), 4 (middle), 5 (bottom)

7.4 Conclusion

In this section we have used all the methods presented in the previous chapters, i.e. the dynamic anisotropic mesh adaptation and the Immerse Volume Method as well as the stabilized finite elements methods to solve coupled industrial problems. The simulation of such processes is essential for industrials in order to understand them better as well as to predict the temperature history in the workpiece and thus optimize these processes. In order to show the performance of the numerical methods, we have simulated three different industrial applications: the cooling of a hat-shaped disc, the heating of a cylindrical ingot in a circular industrial furnace and the quenching of a tubular workpiece in water. We have been able to simulate the three processes entirely, which is already a crucial improvement given the complexity of the cases. The solvers have demonstrated that they are able to provide stable solutions for high Reynolds and Peclet numbers, and the anisotropic mesh adaptation has shown to be extremely useful and efficient to avoid the use of a drastic number of elements and keep a good accuracy.

The numerical results obtained for the first case, which is the cooling of the disc by natural convection are in good accordance with the experimantal data. We have shown that it is essential to adapt the mesh on the temperature and on the level-set of the solid as well, in order to keep an accurate interface and consequently get proper mixing laws. The numerical results of the two other cases are less accurate. In fact these processes being more complex in terms of physics, further numerical models are needed in order to get more accurate results. Therefore further developments are needed to take into account the combustion of gas, a better radiation model and a boiling model. Nevertheless we have pointed out the effeciency and the usefulness of the anisotropic mesh adaptation method which does not imply an increase in the computational time and provides more accurate results than fixed uniform meshes. Therefore we have shown that with the coupling of all the numerical methods we are able to simulate full industrial processes.

Résumé français

La simulation numérique des procédés tels que la trempe ou le chauffage de pièces métalliques dans des fours est essentiel pour les industriels. Ces procédés étant complexes de part les géométries et la variété des phénomènes physiques mis en jeu, il est difficile, cher, et parfois impossible de les analyser et les comprendre en profondeur par le biais d'essais expérimentaux. Ainsi les méthodes numériques ont naturellement pris une place privilégiée pour prédire la physique et mieux maîtriser ces procédés. Dans de tels procédés il est essentiel de bien contrôler l'histoire thermique des pièces métalliques afin d'optenir la miscrostructure désirée, et donc des pièces de qualités en vue de leur future utilisation.

Dans ce chapitre on utilise les méthodes numériques présentées dans les chapitres précédents, à savoir l'adaptation dynamique de maillage anisotrope, la méthode d'immersion de volume et les méthodes d'éléments finis stabilisées pour résoudre des problèmes industriels couplés. Trois procédés industriels sont simulés afin de tester la performance des outils numériques: le refroidissement d'un disque par convection naturelle, le chauffage d'un lingot cylindrique dans un four industriel circulaire et enfin la trempe d'un tube dans l'eau. Ces trois procédés sont très utilisés par les industriels et mettent en jeu des phénomènes physiques différents. Les trois procédés ont été simulé numériquement dans leur intégralité, ce qui apporte déjà une avancée majeure étant donnée la complexité des cas traités. Les solveurs éléments finis ont démontré leur capacité à calculer des solutions stables malgré des nombres de Reynolds et de Peclet élevés, et la méthode d'adaptation de maillage anisotrope s'est avérée efficace et extrèmement utile pour réduire le nombre d'éléments dans les simulations tout en gardant une bonne précision.

Les résultats numériques du premier cas (refroidissment du disque par convection naturelle) sont proches des résultats expérimentaux. Nous avons montré qu'il est essentiel d'adapter le maillage sur la température mais également à l'interface du solide afin de bien capturer cette dernière et par conséquent obtenir des lois de mélange régulières. Les résultas numériques des deux autres cas sont plus contrastés. En effet ces deux procédés étant physiquement plus complexes, des modèles numériques supplémentaires sont nécessaires pour obtenir des résultats plus précis. Ainsi la prise en compte de la combustion des gazs ou l'ébullition fera l'objet de futurs développements. Une amélioration du modèle de rayonnement est également nécessaire. Néanmoins nous avons montré l'efficacité et l'utilité de la méthode d'adaptation de maillage qui n'implique pas une augmentation du temps de calcul et permet de simuler de tels procédés en obtenant des résultas plus précis que des maillages isotropes fixes. Ainsi, avec le couplage de toutes ces méthodes numériques, il est possible de simuler l'intégralité de procédés industriels complexes.

Chapter 8

Conclusion & Perspectives

The objective of this thesis is to make the simulation of real industrial heat treatment processes more realistic. By realistic, we mean reasonable computational time and real configurations (complex and large domains). The software Thost has initially been created to simulate numerically such processes. The software has shown to handle accurately complex coupled turbulent and heat transfer problems. Nevertheless the high resolution meshes needed to capture the physics of such cases are definitely very expensive. Therefore the target of this work is to deal with the computational time and at the same time increase the accuracy of the simulations.

As a first step to improve the accuracy we have proposed a new immersed method. Usually, when dealing with complex geometries, the solids are immersed in the computational domain by computing the level-set to their surface mesh file. Such an approach minimizes the potential of anisotropic mesh adaptation. Indeed when using anisotropic mesh adaptation, the mesh is extremely refined along the fluid-solid interface, until a certain point where the computational mesh recovers the edges induced by the surface mesh file of the solid. Therefore the accuracy of the fluid-solid interface is directly dependant on the quality of the surface mesh file of the solid. Hence, in Chapter 3 we propose a new method to immerse directly the analytical geometry of the solids. We compute the level-set relatively to the CAD file of the objects, containing NURBS functions.

Inspired by the litterature, we have developed different methods in order to compute the distance relatively to NURBS functions. These methods mainly consists in projecting a point on a NURBS curve or surface by finding a good initial guess value in the view of the use of an iterative method. The initial guess value can be found by splitting the NURBS into sub-parts or sample the NURBS into a reasonable number of points. Then different iterative methods have been developed, and we have shown that the faster robust method is the Newton-Raphson method. All the iterative methods as well as the basic tools of NURBS have been presented in Chapter 2.

Afterwards in Chapter 4, we have combined the new immersion method based on NURBS with the presented anisotropic mesh adaptation method in order to solve CFD problems. The method has provided accurate results but further investigations are needed to reduce the computational time. Therefore alternative methods have been presented. The first one transfers the NURBS-based level-set from an optimized mesh to the computational mesh, leading to a important reduction of the computational time. The second one immerses point clouds. Even if this method needs to be improved, it has a good potential considering the continual evolution of lasers and geometries described by point clouds.

Then in the second part of the thesis, we have presented stabilized finite element methods in Chapter 5 for coupling turbulent flows and heat transfers. In the case of convection-dominated problems, the standard Galerkin finite elements method fails at solving these problems and spurious oscillations can appear. The use stabilized finite element methods prevent these numerical oscillations. To demonstrate the capability of the stabilized solvers to handle complex cases, we have tested them on an industrial application which is the heating of six ingots in a furnace. We have also shown that the use of anisotropic mesh adaptation is crucial in order to recover the second order of convergence lost by the stabilized finite elements methods, but also to recover the accuracy of the Immersed NURBS Method.

The anisotropic mesh adaptation method has been presented in Chapter 6. The method is based on an edge-based error estimator, leading to stretching factors and metrics computations. The method is practical because it is based on a specified fixed number of nodes, and it allows to adapt the mesh on multiple criteria without doing any metric intersection. These criteria can be the solutions of the PDEs of the problems, for example the velocity or the temperature. The anisotropic mesh adaptation has been tested on 2D cases and benchmarks and has shown a good potential to simulate accurately complex problems with a reduced number of nodes while keeping a good accuracy.

Finally in Chapter 7 we have used all the presented methods to solve real industrial applications. With such methods it is now possible to simulate numerically entire indutrial processes like cooling by natural convection, heating in furnaces or quenching. Clearly, a number of other considerations have to be taken into account for more accurate predictions of temperature profiles in the furnace chamber and in quenching processes. Hence, here is a list of important steps to enhance the simulation tools for more realistic problems:

- radiation: today we are using a P1 model known to be a diffusive. An extension of this model taking into account priviledged directions for radiation is considered in [72]. Therefore, radiation from the hot gas and the walls of the furnace will be absorbed properly by the conductive solid. Combined with the developed methods in this thesis, the evolution of the temperature inside the furnace will be accurately computed.

- combustion: for the time being, combustion was not considered. Empirical computations are provided by our industrial partners to apply simple boundary conditions (velocity and temperature) for the burners. A better determination of the temperature and the velocity must be considered throughout the simulation using a direct computation of combustion models.

- boiling: for accurate prediction of temperature profile in the quenching processes, the boiling phenomena must be taken into account. Indeed, the phase change, the creation of vapor films and turbulent multi-phase flows are important ingredients for more realistic quenching processes.

Résumé français

L'objectif de cette thèse est de rendre les simulations numériques de procédés industriels plus réalistes, c'est à dire rendre les temps de calcul plus raisonnables et donc rendre les simulations plus exploitables pour les industriels. Le logiciel Thost a été créé initialement pour simuler de tels procédés. Le logiciel a montré sa capacité à résoudre des problèmes complexes couplant écoulements turbulents et transferts thermiques. Cependant la finesse des maillages imposée par la complexité des cas implique des temps de calcul élevés. Ainsi le but de ce travail est de réduire les temps de calcul et/ou d'améliorer la précision des calculs.

Dans la première partie de cette thèse nous avons proposé une méthode d'immersion innovante. Habituellement, lorsque les géométries sont complexes, les solides sont immergés en calculant la fonction distance signée relativement à leur maillage surfacique. Une telle approche diminue le potentiel de la méthode d'adaptation de maillage anisotrope. En effet, le maillage du domaine de calcul peut être adapté si finement à l'interface solide-liquide que sa taille de maille peut devenir inférieure à celle du maillage surfacique de l'objet, révélant ainsi sa facétisation. Ainsi la précision de la fonction level-set est directement dépendante de la qualité du maillage surfacique initial. Nous proposons donc une nouvelle méthode pour immerger les solides directement à partir de leur géométrie analytique, c'est à dire à partir de leur fichiers CAO. Plusieurs méthodes ont été développées pour calculer la fonction distance relativement à des NURBS. Ces méthodes consistent principalement à projeter un point sur les NURBS en trouvant un point de départ judicieux en vue de la résolution par une méthode itérative. Ces méthodes ont été testées sur des cas complexes et ont montré un bon potentiel en terme de précision. Cependant les temps de calculs demeurent longs et nécessitent une amélioration des méthodes.

Dans la seconde partie nous avons présenté la méthode d'adaptation de maillage anisotrope. La méthode est originale car basée sur un estimateur d'erreur sur les arrêtes du maillage et elle permet de considérer plusieurs critères d'adaptation tout en s'affranchissant de calculs d'intersection de métriques. Nous avons montré l'étendue du potentiel de la méthode pour résoudre des problèmes de manières précise tout en diminuant fortement le nombre de noeuds du maillage sur un cas 2D et un benchmark.

Afin de résoudre des problèmes à nombres de Reynolds et de Peclet élevés, nous avons présenté des méthodes éléments finis stabilisées. En effet dans le cas de problèmes à convection dominante, des oscillations numériques peuvent apparaître. Ces méthodes stabilisées permettent de s'affranchir de ces oscillations. La robustesse des méthodes a été démontré sur un cas de fours de industriel.

Enfin le couplage des méthodes numériques présentées a été mis à l'épreuve sur plusieurs cas industriels. Avec de telles méthodes il est à présent possible de simuler l'intégralité

des procédés industriels, ce qui nécéssitaient des temps de calculs trop important auparavant. Cependant des développements supplémentaires sont necéssaires pour améliorer la précision des modèles, comme l'amélioration du modèle de radiation, l'ajout d'un modèle de combustion et d'un modèle d'ébullition.

Bibliography

[1] D. P. Mok and W. A. Wall. Partitioned analysis schemes for the transient interactions in incompressible flows and non linear flexible structures. In *Trends in computational structural mechanics, CIMNE*, Barcelona, 2001.

[2] P. Le Tallec and J. Mouro. Fluid structure interaction with large structural displacements. *Computer Methods in Applied Mechanics and Engineering*, 190(24-25): 3039–3067, 2001.

[3] W. A. Wall, D. P. Mok, and E. Ramm. Partitioned analysis approach for the transient coupled response of viscous fluids and flexible structures. In *ECCM'99*. European conference on computational mechanics, August31 -September 3 1999.

[4] M. A. Fernández and M. Moubachir. A Newton method using exact Jacobians for solving fluid-structure coupling. *Computers and Structure*, 83(2-3):127–142, 2005.

[5] J-F. Gerbeau and M. Vidrascu. A quasi-Newton algorithm based on a reduced model for fluid structure interaction problems in blood flow. *Mathematical Modelling and Numerical Analysis*, 37:631–647, 2003.

[6] J-F. Gerbeau, M. Vidrascu, and P. Frey. Fluid structure interaction in blood flows on geometries coming from medical imaging. *Computers and Structure*, 83(2-3): 155–165, 2005.

[7] P. Causin, J.-F. Gerbeau, and Nobile F. Added-mass effect in the design of partioned algorithms for fluid-structure problems. *Computer Methods in Applied Mechanics and Engineering*, 194:4506–4527, 2005.

[8] K. J. Bathe, H. Zhang, and M. H. Wang. Finite element analysis of incompressible and compressible fluid flows with free surfaces and structural interactions. *Computers and Structures*, 56:193–213, 1995.

[9] K. J. Bathe and H. Zhang. A flow-condition-based interpolation finite element procedure for incompressible fluid flows. *Computers and Structures*, 80:1267–1277, 2002.

[10] C. Michler, EH. van Brummelen, SJ. Hulshoff, and R. de Borst. A monolithic approach to fluid structure interaction. *Computers and Fluids*, 33:839–848, 2004.

[11] HO. Kreiss and A. Petersson. A second-order accurate embedded boundary method for the wave equation with dirichlet data. *SIAM Journal on Scientific Computation*, 27:1141–1167, 2006.

[12] CS. Peskin. Flow patterns around heart valves: a numerical method. *Journal of Computational Physics*, 10:252–271, 1972.

[13] R. Glowinski, TW. Pan, AJ. Kearsley, and J. Periaux. Numerical simulation and optimal shape for viscous flow by a fictitious domain method. *International Journal for Numerical Methods in Fluids*, 20:695–711, 2005.

[14] E. Hachem, H. Digonnet, E. Massoni, and T. Coupez. Immersed volume method for solving natural convection, conduction and radiation of a hat-shaped disk inside a 3d enclosure. *International Journal of Numerical Methods for Heat & Fluid Flow*, 22 (6):718–741, 2012.

[15] H. Johansen and P. Colella. A cartesian grid embedded boundary method for poisson's equation on irregular domains. *Journal of Computational Physics*, 147: 60–85, 1998.

[16] C. Farhat, A. Rallu, K. Wang, and T. Belytschko. Robust and provably second-order explicit-explicit and implicit-explicit staggered time-integratorsfor highly nonlinear fluid-structure interaction problems. *International Journal for Numerical Methods in Engineering*, 84(1):73–107, 2010.

[17] C. Farhat, K. Maute, B. Argrow, and M. Nikbay. A shape optimization methodology for reducing the sonic boom initial pressure rise. *AIAA Journal of Aircraft*, 45:1007–1018, 2007.

[18] F. Ilinca and J.-F. Hetu. A finite element immersed boundary method for fluid flow around rigid objects. *International Journal for Numerical Methods in Fluids*, 65:856–875, 2011.

[19] H. Beaugendrea R. Abgrall and C. Dobrzynskia. An immersed boundary method using unstructured anisotropic mesh adaptation combined with level-sets and penalization techniques. *Journal of Computational Physics*, 257:83–101, 2014.

[20] D-L. Quan, T. Toulorge, E. Marchandise, J-F. Remacle, and G. Bricteux. Anisotropic mesh adaptation with optimal convergence for finite elements using embedded geometries. *Computer Methods in Applied Mechanics and Engineering*, 268:65–81, 2014.

[21] E. Hachem. *Stabilized Finite Element Method for Heat Transfer and Turbulent Flows inside Industrial Furnaces.* PhD thesis, Ecole Nationale Supérieure des Mines de Paris, 2009.

[22] H. Digonnet, S. Luisa, and T. Coupez. Cimlib: A fully parallel application for numerical simulations based on components assembly. pages 269–274. Proceedings of the 9th International Conference on Numerical Methods in Industrial Forming Processes, 2007.

[23] S. Balay, K. Buschelman, V. Eijkhout, W.D. Gropp, D. Khaushig, M.G. Knepley, L.C. McInnes, B.F. Smith, and H. Zhang. Petsc users manual. *Technical Report ANL-95/11 - Revision 3.0.0, Argonne National Laboratory*, 2008.

[24] T. Coupez, H. Digonnet, and R. Ducloux. Parallel meshing and remeshing. *Applied Mathematical*, 25:153–175, 2000.

[25] J. Bruchon, H. Digonnet, and T. Coupez. Using a signed distance function fort he simulation of metal forming processes: formulation of the contact condition and mesh adaptation. from a lagrangian approach to an eulerian approach. *International Journal for Numerical Methods in Engineering*, 78:980–1008, 2009.

[26] R. Codina and O. Soto. A numerical model to track two-fluid interfaces based on a stabilized finite element method and the level set technique. *International Journal for Numerical Methods in Fluids*, 40:293–301, 2002.

[27] S.P. van der Pijl, A. Segal, C. Vuik, and P. Wesseling. A mass-conserving level-set method for modelling of multi-phase flows. *International Journal for Numerical Methods in Fluids*, 47:339–361, 2005.

[28] J. Dompierre, M. G. Vallet, M. Fortin, W. G. Habashi, S. Boivin, Y. Bourgault, and A. Tam. Edge-based mesh adaptation for cfd. international conference on numerical methods for the euler and navier-stokes equations. In *8th IEEE Symp. pn Parallel and Distributed Processing*, pages 265–299, Montréal, Sept. 1995.

[29] P. J. Frey and F. Alauzet. Anisotropic mesh adaptation for cfd computations. *Computer Methods in Applied Mechanics and Engineering*, 194(48-49):5068–5082, 2005.

[30] J.-F. Remacle, X. Li, M.S. Shephard, and J.E. Flaherty. Anisotropic adaptive simulation of transient flows using discontinuous galerkin methods. *International Journal for Numerical Methods in Engineering*, 62:899–923, 2005.

[31] C. Gruau and T. Coupez. 3D tetrahedral, unstructured and anisotropic mesh generation with adaptation to natural and multidomain metric. *Computer Methods in Applied Mechanics and Engineering*, 194:4951–4976, 2005.

[32] T. Coupez. Génération de maillage et adaptation de maillage par optimisation locale. *Revue européenne des éléments finis*, 9:403–423, 2000.

[33] Y. Mesri, H. Digonnet, and T. Coupez. Advanced parallel computing in material forming with cimlib. *European Journal of Computational Mechanics*, 18(7-8):669–694, 2009.

[34] T. Coupez. Metric construction by length distribution tensor and edge based error for anisotropic adaptive meshing. *Journal of Computational Physics*, 230: 2391–2405, 2011.

[35] T. Coupez, J. Jannoun, N. Nassif, H.C. Nguyen, H. Digonnet, and E. Hachem. Adaptive time-step with anisotropic meshing for incompressible flows. *Journal of Computational Physics*, 241:195 – 211, 2013.

[36] S-V. Patankar. *Numerical Heat Transfer and Fluid Flow.* Series in Computational and Physical Processes in Mechanics and Thermal Sciences. Taylor & Francis, 1980.

[37] T. Hughes, J.A. Cottrell, and Y. Bazilevs. Isogeometric analysis: Cad, finite elements, nurbs, exact geometry and mesh refinement. *Computer Methods in Applied Mechanics and Engineering*, 194:4135–4195, 2005.

[38] L.A. Piegl, K. Rajab, Smarodzinava. V, and K.P. Valavanis. Point-distance computations: A knowledge-guided approach. *Computer-Aided Design and Applications*, 5(6):855–866, 2008.

[39] L-A. Piegl and W. Tiller. *The NURBS Book 2nd Edition.* Springer, 1996.

[40] P. De Casteljau. Outillages methodes calcul. Technical report, A. Citroen, Paris, 1959.

[41] P. Bezier. Definition numerique des courbes et surfaces i. *Automatisme*, XI:625–632, 1966.

[42] MG. Cox. The numerical evaluation of b-splines. Technical report, National Physics Laboratory DNAC4, 1971.

[43] C. De Boor. On calculation with b-splines. *Journal of Approximation Theory*, 6: 50–62, 1972.

[44] G. Elber. *Free form surface analysis using a hybrid of symbolic and numeric computations.* PhD thesis, University of Utah, 1992.

[45] L-A. Piegl and W. Tiller. Symbolic operators for nurbs. *Computer-Aided Design*, 29(5):361–368, 1997.

[46] E. Dyllong and W. Luther. Distance calculation between a point and a nurbs surface. In *Curve and Surface Design, Saint-Malo, 55-62*, 1999.

[47] Y.L. Ma and W.T. Hewitt. Point inversion and projection for nurbs curve and surface: Control polygon approach. *Computer Aided Geometric Design*, 20:79–99, 2003.

[48] X.D. Chen, H. Su, J.H. Yong, J.C. Paul, and J.G. Sun. A counterexample on point inversion and projection for nurbs curve. *Computer Aided Geometric Design*, 24: 302, 2007.

[49] I. Selimovic. Improved algorithms for the projection of points on nurbs curves and surfaces. *Computer Aided Geometric Design*, 23:439–445, 2006.

[50] X.D. Chen, J.H. Yong, G. Wang, J.C. Paul, and G. Xu. Computing the minimum distance between a point and a nurbs curve. *Computer-Aided Design*, 40:1051–1054, 2008.

[51] Y-t. Oh, Y-J. Kim, J. Lee, M-S. Kim, and G. Elber. Efficient point-projection to freeform curves and surfaces. *Computer Aided Geometric Design*, 29:242–254, 2012.

[52] S-M. Hu and J. Wallner. A second order algorithm for orthogonal projection onto curves and surfaces. *Computer Aided Geometric Design*, 22:251–260, 2005.

[53] P. Schneider and D. Eberly. *Geometric Tools for Computer Graphics*. Morgan Kaufmann Publishers, 2003.

[54] W. Press, S. Teukolsky, Vetterling W., and Flannery B. *Numerical Recipes, The Art of Scientific Computing, 3rd Edition*. Cambridge University Press, 2007.

[55] H-C. Song, X. Xu, K-L. Shi, and J-H. Yong. Projecting points onto planar parametric curves by local biarc approximation. *Computers and Graphics*, 38:183–190, 2014.

[56] D.A. Bircan. *Development of a NURBS based adaptive slicing procedure for fused deposition modeling in rapid prototyping applications*. PhD thesis, Cukurova University, 2008.

[57] S. Osher and J. Sethian. Fronts propagating with curvature-dependent speed: algorithms based on hamilton-jacobi formulations. *Journal of Computational Physics*, 79:12–49, 1988.

[58] M.W. Jones and J.A. Baerentzen. 3d distance fields: A survey of techniques and applications. *IEEE Transactions on Visualization and Computer Graphics*, 12(4): 581–599, 2006.

[59] R. Elias, M. Martins, and A. Coutonho. Simple finite element-based computation of distance functions in unstructured grids. *International Journal for Numerical Methods in Engineering*, 72:1095–1110, 2007.

[60] X.D. Chen, G. Xu, J.H. Yong, G. Wang, and J.C. Paul. Computing the minimum distance between a point and a clamped b-spline surface. *Graphical Models*, 71: 107–112, 2009.

[61] E. Hachem, B. Rivaux, T. Kloczko, H. Digonnet, and T. Coupez. Stabilized finite element method for incompressible flows with high reynolds number. *Journal of Computational Physics*, 229:8643–8665, 2010.

[62] Y. Mesri, H. Digonnet, and T. Coupez. Advanced parallel computing in material forming with cimlib. *European Journal of Computational Mechanics*, 18(7-8):669–694, 2009.

[63] L. Kobbelt and M. Botsch. A survey of point-based techniques in computer graphics. *Computer and Graphics*, 28:801–814, 2004.

[64] H. Edelsbrunner. Shape reconstruction with delaunay complex. *Theoritical Informatics*, 1380:119–132, 1998.

[65] D. Rogers. *An introduction to NURBS*. Morgan Kaufmann, 2003.

[66] H. Zhao, S. Osher, and R. Fedwik. Fast surface recontruction using the level set method. In *IEEE Workshop*, pages 194–201, 2001.

[67] O. Schall and M. Samozino. Surface from scattered points: a brief survey of recent developments. *1st International Workshop on Semantic Virtual Environments*, pages 138–147, 2005.

[68] B. E. Launder and D. B. Spalding. The numerical computation of turbulent flows. *Computer Methods in Applied Mechanics and Engineering*, 3(2):269–289, 1974.

[69] W. P. Jones and B. E. Launder. The prediction of laminarization with a two-equation model of turbulence. *International Journal of Heat and Mass Transfer*, 15(2):301–314, 1972.

[70] Michael F. Modest. *Radiative Heat Transfer*. McGraw-Hill, New-York, 1993.

[71] Robert Siegel and John Howell. *Thermal Radiation Heat Transfer*. Taylor & Francis, New-York, 2002.

[72] Q. Schmid. *Stabilized finite elements method for solving radiative transfer in industrial furnaces*. PhD thesis, Mines ParisTech, 2013-2016.

[73] Z. Han and R. D. Reitz. A temperature wall function formulation for variable-density turbulent flows with application to engine convective heat transfer modeling. *International Journal of Heat and Mass Transfer*, 40(3):613–625, 1997.

[74] E. Hachem, B. Rivaux, T. Klozcko, H. Digonnet, and Coupez T. Stabilized finite element method for incompressible flows with high reynolds number. *Journal of Computational Physics*, 229:8643–8665, 2010.

[75] F. Brezzi and J. Douglas. Stabilized mixed methods for the Stokes problem. *Numerische Mathematik*, 53:225–236, 1988.

[76] F. Brezzi and J. Pitkaranta. On the stabilization of finite element approximations of the Stokes problem. *Efficient Solutions of Elliptic Systems, Notes on Numerical Fluid Mechanics*, 10:11–19, 1984.

[77] L.P. Franca and T.J.R. Hughes. Two classes of mixed finite element methods. *Computer Methods in Applied Mechanics and Engineering*, 69:89–129, 1988.

[78] Ramon Codina, José M. González-Ondina, Gabriel Díaz-Hernández, and Javier Principe. Finite element approximation of the modified boussinesq equations using a stabilized formulation. *International Journal for Numerical Methods in Fluids*, 57:1305–1322, 2008.

[79] Ramon Codina. Comparison of some finite element methods for solving the diffusion-convection-reaction equation. *Computer Methods in Applied Mechanics and Engineering*, 156:185–210, 1998.

[80] A.N. Brooks and T.J.R. Hughes. Streamline upwind/Petrov-Galerkin formulations for convection dominated flows with particular emphasis on the incompressible Navier-Stokes equations. *Computer Methods in Applied Mechanics and Engineering*, 32:199–259, 1982.

[81] D.N. Arnold, F. Brezzi, and M. Fortin. A stable finite element for the Stokes equations. *Calcolo*, 23(4):337–344, 1984.

[82] F. Brezzi, M.O. Bristeau, L.P. Franca, M. Mallet, and G. Roge. A relationship between stabilized finite element methods and the Galerkin method with bubble functions. *Computer Methods in Applied Mechanics and Engineering*, 96:117–129, 1992.

[83] R.E. Bank and B.D. Welfert. A comparison between the mini element and the Petrov-Galerkin formulations for the generalized Stokes problem. *Computer Methods in Applied Mechanics and Engineering*, 83:61–68, 1990.

[84] A. Masud and R. A. Khurram. A multiscale finite element method for the incompressible navier-stokes equations. *Computer Methods in Applied Mechanics and Engineering*, 195:1750–1777, 2006.

[85] E. Hachem, B. Rivaux, T. Kloczko, H. Digonnet, and T. Coupez. Stabilized finite element method for incompressible flows with high reynolds number. *Journal of Computational Physics*, 229(23):8643–8665, 2010.

[86] T. J. R. Hughes, G. R. Feijoo, L. Mazzei, and J. N. Quincy. The variational multiscale method - a paradigm for computational mechanics. *Computer Methods in Applied Mechanics and Engineering*, 166:3–24, 1998.

[87] T. Dubois, F. Jauberteau, and R. Temam. *Dynamic Multilevel Methods and the Numerical Simulation of Turbulence*. Cambridge University Press, Cambridge, 1999.

[88] Ramon Codina and Javier Principe. Dynamic subscales in the finite element approximation of thermally coupled incompressible flows. *International Journal for Numerical Methods in Fluids*, 54:707–730, 2007.

[89] Tayfun E. Tezduyar and Yasuo Osawa. Finite element stabilization parameters computed from element matrices and vectors. *Computer Methods in Applied Mechanics and Engineering*, 190(3-4):411–430, 2000.

[90] Ramon Codina. Stabilization of incompressibility and convection through orthogonal sub-scales in finite element methods. *Computer Methods in Applied Mechanics and Engineering*, 190(13-14):1579–1599, 2000.

[91] S. Mittal. On the performance of high aspect ratio elements for incompressible flows. *Computer Methods in Applied Mechanics and Engineering*, 188:269–287, 2000.

[92] S. Micheletti, S. Perotto, and M. Picasso. Stabilized finite elements on anisotropic meshes: A priori error estimates for the advection-diffusion and the stokes problems. *SIAM Journal on Numerical Analysis*, 41:1131–1162, 2004.

[93] I. Harari and T. J. R. Hughes. What are c and h?: inequalities for the analysis and design of finite element methods. *Computer Methods in Applied Mechanics and Engineering*, 97:157–192, 1992.

[94] Santiago Badia and Ramon Codina. Analysis of a stabilized finite element approximation of the transient convection-diffusion equation using an ale framework. *Journal on Numerical Analysis*, 44:2159–2197, 2006.

[95] T.J.R. Hughes, L.P. Franca, and M. Balestra. A new finite element formulation for computational fluid dynamics: V. circumventing the Babuska-Brezzi condition: A stable Petrov-Galerkin formulation of the Stokes problem accommodating equal-order interpolations. *Compter Methods in Applied Mechanics and Engineering*, 59: 85–99, 1987.

[96] A.C. Galeão and E.G.D. do Carmo. A consistent approximate upwind Petrov-Galerkin method for convection-dominated problems. *Computer Methods in Applied Mechanics and Engineering*, 68(1):83–95, 1988.

[97] Elie Hachem. *Stabilized Finite Element Method for Heat Transfer and Turbulent Flows inside Industrial Furnaces*. PhD thesis, Ecole Nationale Supérieure des Mines de Paris, 2009.

[98] T.E. Tezduyar and Y.J. Park. Discontinuity-capturing finite element formulations for nonlinear convection-diffusion-reaction equations. *"Computer Methods in Applied Mechanics and Engineering*, 59(3):307–325, 1986.

[99] Hugues Digonnet and Thierry Coupez. Object-oriented programming for fast and easy development of parallel applications in forming processes simulation. In *Computational Fluid and Solid Mechanics 2003*, pages 1922–1924, 2003.

[100] H. Nguyen, M. Gunzburger, L. Ju, and J. Burkardt. Adaptive anisotropic meshing for steady convection-dominated problems. *Computer Methods in Applied Mechanics and Engineering*, 198 (37-40):2964–2981, 2009.

[101] Z. Zang. Finite element superconvergence on shishkin mesh for 2d convection-diffusion problems. *Mathematics of Computation*, 72 (243):1147–1177, 2003.

[102] E. Hachem, G. Jannoun, J. Veysset, and T. Coupez. On the stabilized finite element method for steady convection-dominated problems with anisotropic mesh adaptation. *Applied Mathematics and Computation*, 232:581–594, 2014.

[103] W. Heiligenstaedt. *Thermique Appliquee aux Fours Industriels*. Paris: Dunod, 1971.

[104] F. Alauzet. Size gradation control of anisotropic meshes. *Finite Elements inanalysis and Design*, 46:181–202, 2010.

[105] J. Peraire, M. Vahdati, K. Morgan, and O.C. Zienkiewicz. Adaptive remeshing for compressible flow computations. *Journal of Computational Physics*, 72(2):449 – 466, 1987.

[106] V. Selmin and L. Formaggia. Simulation of hypersonic flows on unstructured grids. *International Journal for Numerical Methods in Engineering*, 34(2):569–606, 1992.

[107] R. Lohner. Adaptive remeshing for transient problems. *Computer Methods in Applied Mechanics and Engineering*, 75:195–214, 1989.

[108] O. C. Zienkiewicz and J. Wu. Automatic directional refinement in adaptive analysis of compressible flows. *International Journal for Numerical Methods in Engineering*, 37(13):2189–2210, 1994.

[109] Thierry Coupez. A mesh improvement method for 3D automatic remeshing. Numerical Grid Generation in Computational Fluid Dynamics and Related Fields, pages 615–626. Pineridge Press, 1994.

[110] X Li, M. S. Shephard, and M. W. Beall. 3d anisotropic mesh adaptation by mesh modification. *Computer Methods in Applied Mechanics and Engineering*, 194:4915–4950, 2005.

[111] M. S. Shephard, J. E. Flaherty, K. E. Jansen, X Li, X Luo, N. Chevaugeon, J. F. Remacle, M. W. Beall, and R. M. Obara. Adaptive mesh generation for curved-domains. *Applied Numerical Mathematics*, 52:251–271, 2005.

[112] L. Formaggia and S. Perotto. New anisotropic a priori error estimate. *Numer. Math.*, 89:641–667, 2001.

[113] A. Tam, D. Ait-Ali-Yahia, M.P. Robichaud, M. Moore, V. Kozel, and W.G. Habashi. Anisotropic mesh adaptation for 3d flows on structured and unstructured grids. *Computer Methods in Applied Mechanics and Engineering*, 189(4): 1205–1230, 2000.

[114] C.C. Pain, A.P. Umpleby, C.R.E. de Oliveira, and A.J.H. Goddard. Tetrahedral mesh optimisation and adaptivity for steady state and transient finite element calculations. *Computer Methods in Applied Mechanics and Engineering*, 190:3771–3796, 2001.

[115] G. Kunert. An a posteriori residual error estimator for the finite element method on anisotropic tetrahedral meshes. *Numerische Mathematik*, 86(3):471–490, 2000.

[116] L. Formaggia, S. Micheletti, and S. Perotto. Anisotropic mesh adaptation in computational fluid dynamics: Application to the advection-diffusion-reaction and the stokes problems. *Applied Numerical Mathematics*, 51(4):511 – 533, 2004.

[117] S. Micheletti and S. Perotto. Reliability and efficiency of an anisotropic zienkiewicz zhu error estimator. *Computer Methods in Applied Mechanics and Engineering*, 195:799–835, 2006.

[118] M. Picasso. Adaptive finite elements with large aspect ratio based on an anisotropic error estimator involving first order derivatives. *Computer Methods in Applied Mechanics and Engineering*, 196:14–23, 2006.

[119] F. Hecht and R. Kuate. An approximation of anisotropic metrics from higher order interpolation error for triangular mesh adaptation. *J. Comput. Appl. Math.*, 258: 99–115, 2014. ISSN 0377-0427.

[120] Weizhang Huang. Metric tensors for anisotropic mesh generation. *J. Comput. Phys.*, 204(2):633–665, 2005.

[121] D.A Venditti and D. L. Darmofal. Anisotropic grid adaptation for functional outputs: application to two-dimensional viscous flows. *Journal of Computational Physics*, 187(1):22 – 46, 2003.

[122] S. Micheletti and S. Perotto. Output functional control for nonlinear equations driven by anisotropic mesh adaption: The navier-stokes equations. *SIAM Journal on Scientific Computing*, 30(6):2817–2854, 2008.

[123] J. Peter, M. Nguyen-Dinh, and P. Trontin. Goal oriented mesh adaptation using total derivative of aerodynamic functions with respect to mesh coordinates. with applications to euler flows. *Computers & Fluids*, 66(0):194 – 214, 2012.

[124] F. Alauset. *Adaptation de maillage anisotrope en trois dimension. Applications aux simulations instationnaires en Mecanique des Fluides*. PhD thesis, Universite de Montpellier II, 2003.

[125] F. Alauzet, P.J. Frey, P.-L. George, and B. Mohammadi. 3d transient fixed point mesh adaptation for time-dependent problems: Application to cfd simulations. *Journal of Computational Physics*, 222:592–623, 2007.

[126] T. Coupez and E. Hachem. Solution of high-reynolds incompressible flow with stabilised finite element and adaptive anisotropic meshing. *Computer Methods in Applied Mechanics and Engineering*, 2012.

[127] T. Coupez, G. Jannoun, J. Veysset, and E. Hachem. Edge-based anisotropic mesh adaptation for cfd applications. *Proceedings of the 21st International Meshing Roundtable*, pages 567–583, 2013.

[128] G. Jannoun. *Space-time adaptive stabilized finite elements method for the resolution long time and large scales applications*. PhD thesis, Mines ParisTech, 2014.

[129] G. De Vahl Davis. Natural convection in a square cavity: a comparison exercise. *International Journal for Numerical Methods in Fluids*, 3:227–248, 1983.

[130] N. C. Markatos and Pericleous K. A. Laminar and turbulent natural convection in an enclosed cavity. *International Journal of Heat and Mass Transfer*, 27:755–772, 1984.

[131] R. Henkes and C. Hoogendoorn. Scaling of the laminar natural-convection flow in a heated square cavity. *International Journal of H*, 36:2913–2925, 1992.

[132] P. Le Quere, C. Weisman, H. Paillere, J. Vierendeels, E. Dick, R. Becker, M. Braack, and J. Locke. Modelling of natural convection flows with large temperature differences: a benchmark problem for low mach number solvers. part 1. reference solutions. *Mathematical Modelling and Numerical Analysis*, 39:609–616, 2005.

[133] H.N. Dixit and V. Babu. Simulation of high rayleigh number natural convection in a square cavity using the lattice boltzmann method. *International Journal of Hea*, 49:727–739, 2006.

Simulation des grands espaces et des temps longs

RÉSUMÉ : L'interaction fluide structure est présente dans beaucoup de problèmes industriels. Même si les performances informatique s'améliorent considérablement et que les méthodes en mécanique numérique gagnent en maturité, certaines difficultés ne permettent pas encore de réaliser des simulations numériques précises.

Différentes approches ont été développée, dont la Méthode d'Immersion de Volume. Cette méthode permet de faciliter la mise en place des calculs. Ainsi il n'est pas nécessaire de construire des maillages concordant avec la géométrie des objets, et le couplage entre les fluides et les solides se fait naturellement.

C'est sur cette analyse qu'à été développé le logiciel Thost. Il permet de simuler des procédés industriels tels que le chauffage de pièces métalliques dans les fours industriels ou la trempe sans caractériser expérimentalement des coefficients de transfert. Cependant les coûts de calcul restant élevés, le but de la thèse est de les diminuer en s'appuyant sur des méthodes numériques innovantes tels que le l'adaptation dynamique de maillage anisotrope, des méthodes éléments finis stabilisées ou l'immersion direct des objets à partir de la Conception Assistée par Ordinateur.

Mots clés : NURBS; Adaptation de maillage anisotrope; Intéraction fluide-structure (IFS); Méthode éléments finis stabilisée; Méthode d'immersion de volume; Conception Assisistée par ordinateur (CAO)

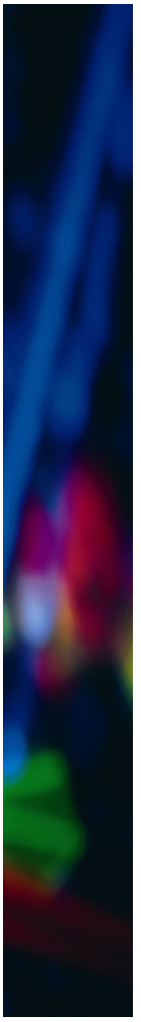
Numerical modeling of large scales and long time

ABSTRACT : Fluid-Structure Interaction (FSI) describes a wide variety of industrial problems. In spite of the available computer performance and the actual maturity of computational fluid dynamics and computational structural dynamics, several key issues still prevent accurate FSI simulations.

Two main approaches for the simulation of FSI problems are still gaining attention lately: partitioned and monolithic approaches. Monolithic methods are still of interest due to their capability to treat the interaction of the fluid and the structure using a unified formulation. In fact it makes the build up of a FSI problem easier as the mesh do not have to fit the geometry of the solids and the transfers are treated naturally.

The software Thost has been created based on these analyzes. Thost is a 3D aerothermal numerical software. Its target is to model numericaly the thermal history of the industrial pieces in their environment without using any transfer coefficient. However the computational costs are still high and therefore the software is not fully efficient from an industrial point of view to simulate, analyze and improve complex processes. All the work in this PhD thesis has been done to reduce the computational costs and optimize the accuracy of the simulations in Thost based on innovative numerical methods such as dynamic anisotropic mesh adaptation, stabilized finite elements methods and immersing the objects directly from their Computer Aided Design files.

Key words : NURBS; Anisotropic mesh adaptation; Fluid-structure interaction (FSI); Stabilized finite elements method; Immersed volume method; Computer aided design (CAD)



MINES ParisTech